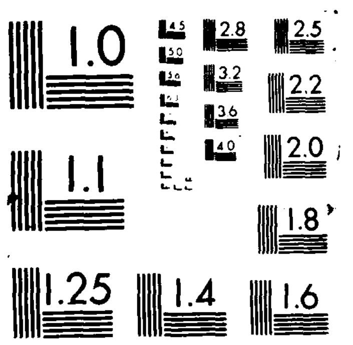
AD-A186 305 A PROTOTYPE DECISION SUPPORT SYSTEM FOR MARINE CORPS OFFICER ALLOCATION POLICY ANALYSIS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA P J EXNER SEP 87 1/2

UNCLASSIFIED F/G 5/1 NL


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DEPARTMENT OF THE NAVY NAVAL POSTGRADUATE SCHOOL

THESIS

A PROTOTYPE DECISION SUPPORT SYSTEM FOR MARINE CORPS OFFICER ALLOCATION POLICY ANALYSIS

by

Philip J. Exner

September 1987

Thesis Advisor: Paul R. Milch

87 11 9 048

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 55	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) A PROTOTYPE DECISION SUPPORT SYSTEM FOR MARINE CORPS OFFICER ALLOCATION POLICY ANALYSIS

12 PERSONAL AUTHOR(S)
EXNER, Philip J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1987 September	150

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Manpower Allocation, Decision Support System, Mathematical Network

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis presents the prototype for a decision support system which permits repeated formulation and solution of the Marine Corps staffing allocation problem under various user-controlled policy scenarios. The system allows the decision maker to vary the eligibility criteria used to determine who may be transferred as well as to adjust the relative priorities of the two objectives: minimize relocation costs and maximize "fit" as defined by the Marine Corps. The user may also set a minimum acceptable "aspiration level" for the total fill of all billets. Based on the eligibility requirements which are input by the user, the system extracts data on individual Marines and the jobs that need to be filled, and matches people to billets using a set of matching rules developed by the Marine Corps. The resulting matches are then transformed into a capacitated transshipment network for solution in a special

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL PROF. PAUL R. MILCH	22b TELEPHONE (Include Area Code) 408-646-2882 22c OFFICE SYMBOL 55Mh

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete   SECURITY CLASSIFICATION OF THIS PAGE

A-1

BLOCK 19, ABSTRACT (cont):

commercial optimization software package. The network formulation models a multiobjective allocation problem using optimization techniques to permit adjustment of some of the objective priorities. After the solution is presented to the decision maker, he may change the relative priorities of the relocation cost and fit objectives, set an aspiration level for the total fill, or change the rules used to determine who may be transferred. The user then has the option of reformulating and re-solving the problem with the new objective priorities or aspirations, or re-starting the entire problem based on the new eligibility rules. Testing of the system on all Marine Corps aviation officers, who constitue about 35 percent of the total personnel, suggests that the concept is feasible to implement for the entire Marine Corps, provided that certain enhancements recommended in the thesis are implemented.

A prototype decision support system for Marine Corps officer allocation policy analysis

by

Philip J. Exner
Captain, United States Marine Corps
B.S., United States Naval Academy, 1976

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

Author: [illegible]
Philip J. Exner

Approved by: Paul R. Milch
Paul R. Milch, Thesis Advisor

[illegible]
G.H. Bradley, Second Reader

P. Purdue.
P. Purdue, Chairman,
Department of Operations Analysis

[illegible]
James M. Fremgen,
Acting Dean of Information and Policy Sciences

ABSTRACT

This thesis presents the prototype for a decision support system which permits repeated formulation and solution of the Marine Corps staffing allocation problem under various user-controlled policy scenarios. The system allows the decision maker to vary the eligibility criteria used to determine who may be transferred as well as to adjust the relative priorities of two objectives: minimize relocation costs and maximize "fit" as defined by the Marine Corps. The user may also set a minimum acceptable "aspiration level" for the total fill of all billets. Based on the eligibility requirements which are input by the user, the system extracts data on individual Marines and the jobs that need to be filled, and matches people to billets using a set of matching rules developed by the Marine Corps. The resulting matches are then transformed into a capacitated transshipment network for solution in a special commercial optimization software package. The network formulation models a multiobjective allocation problem using optimization techniques to permit adjustment of some of the objective priorities. After the solution is presented to the decision maker, he may change the relative priorities of the relocation cost and fit objectives, set an aspiration level for the total fill, or change the rules used to determine who may be transferred. The user then has the option of reformulating and re-solving the problem with the new objective priorities or aspirations, or re-starting the entire problem based on the new eligibility rules. Testing of the system on all Marine Corps aviation officers, who constitute about 35 percent of the total personnel, suggests that the concept is feasible to implement for the entire Marine Corps, provided that certain enhancements recommended in the thesis are implemented.

TABLE OF CONTENTS

I. INTRODUCTION

The United States Marine Corps does not have an integrated system for evaluating the impact of changes in its policies or funding on manpower issues. Presently, any queries regarding the effect of some policy proposal or budgetary constraint on manning levels in the Marine Corps are handled on an ad hoc basis. This involves time-consuming sorting through data files to develop appropriate tests; however the results may not provide a clear or reliable picture of the effect of the proposal on other aspects of manpower allocation and assignment.

The purpose of this thesis is to develop the prototype for a manpower decision support system which can assist Marine Corps policy makers in evaluating the impact on officer staffing of tradeoffs among fill, fit, dollar costs, and changes in various assignment and allocation-related policies. Specifically, the model is designed to demonstrate the feasibility of the concept of using available data bases and software in a flexible network assignment model to measure the effect of changes in policy and budgetary constraints on the ability to staff the Marine Corps to some desired level of fill and fit.

In this chapter, the requirement for the system is described, the concept and scope of the model is explained, and an overview of the solution approach is presented. Finally, a general outline of the structure of the thesis is provided.

A. REQUIREMENT FOR SYSTEM

As the federal deficit continues to command Congressional attention, the pressure for further military spending cuts will continue to grow. Many of these cuts will affect programs such as operational Permanent Change of Station (PCS) personnel moves where a cut does not produce an easily quantifiable reduction in readiness. Notwithstanding, operational PCS moves are considered an important part of the development of a well-rounded, versatile and experienced force, and are an integral part of normal career patterns.

In the Fiscal Year 1987 budget, Congress imposed a spending cut which resulted in the delay and, in some cases, the outright cancellation of numerous PCS moves. Further funding reductions are expected in the years ahead which may compel the Marine Corps to change some of its policies with regard to tour length, overseas moves, and unaccompanied tours. This may make it difficult to achieve required staffing levels at some commands and will affect the ability of the assignment officers, called "monitors", to achieve the desired fit in many billets. Thus it is increasingly important that the Marine Corps continue to improve its management of PCS costs and develop some method for assessing the impact of proposed policy or budgetary changes on staffing and assignments. A comprehensive system for more accurately estimating staffing levels under various policy and budgetary conditions could help to validate PCS budget requests to Congress, as well as enhance the overall efficiency and combat readiness of the Marine Corps.

The manpower allocation models which might be adapted for such a system use a variety of mathematical and heuristic techniques to deal with the large number of objectives that must be considered. The Officer Staffing Goal Model (OSGM), which is presently used by the Marine Corps for determining allocation goals, successively considers total fill, fill within job priority levels, and fit while enforcing a proportionality constraint which causes the equal sharing of shortages among billets of the same priority. However, because of the rigid data structures used in the model and the design of the OSGM itself, it is not suitable for analysis of policy alternatives. Also, since it does not consider relocation or PCS costs, it cannot be used to evaluate the effect of changes in budgetary constraints.

This thesis presents the design for a decision support system which could assist Marine Corps decision makers in the analysis of a broad range of policy questions. Through the use of a special mathematical network formulation, the model considers the impact of several important factors on the fill, fit, and PCS cost of Marine Corps staffing, while observing certain restrictions and matching rules which are used in the actual staffing process. The system uses the same basic input files as the OSGM and would therefore not require the development and maintenance of new data bases. The prototype presented here permits both manipulation of the data extraction rules and adjustment of the network formulation. The user is permitted to modify the rules determining who may be eligible to move, as well as adjust both the minimum acceptable number of billets filled and the weights attached to some of the objectives. By permitting exploration of the optimal staffing levels under various policy and budgetary constraints, the model can provide the decision maker with a tool for better managing the funding and manpower assets of the Marine Corps.

B. CONCEPT AND SCOPE OF THE MODEL

1. Concept

The model is based on the idea that in any major policy area, such as PCS costs, there are "families" of policy-related questions which can be defined with reasonable precision. In addition, questions about policies which can be evaluated quantitatively can usually be answered by reference to a limited number of data bases which contain information relevant to certain question types. By permitting the decision maker to change the rules used to extract and organize the data, or to adjust the constraints and "weights" (either implicit or explicit) associated with the various objective functions, he may explore the outcomes of a process under varying conditions.

The model presented in this paper demonstrates the feasibility of building a decision support system based upon that concept, and is guided by several practical considerations:

1. The model should utilize data bases which already exist and are readily accessible. Data requirements for the model should not necessitate the development of any major new data bases nor require extensive modification or special maintenance of present data.
2. Ideally, the model should not require acquisition of any new hardware, and should be built around existing software when possible. Despite the size and complexity of the problem, the model should not require the use of a dedicated mainframe, and should run fast enough to permit multiple queries in a reasonable period of time.
3. The model should be flexible and adaptable. Flexibility can be measured by the model's ability to answer a wide variety of "what-if" questions regarding PCS-related manpower issues. Adaptability involves the ability to incorporate enhancements or modify existing routines without making major structural changes to the model.
4. The decision maker should be able to modify eligibility rules, set targets for some objectives, and adjust the priorities of some of the goals.

2. Scope

This prototype focuses on the interaction of PCS moving costs and policies on staffing levels among Marine aviation and air support officers (who constitute about 35-40% of Marine Corps officers.) It is designed modularly to facilitate implementation of the full scale model involving all Marine Corps officers, and to simplify future addition of a capability to explore other manpower questions as requirements change. By modularizing the data preparation and problem formulation process, a relatively large problem can be solved without requiring a dedicated mainframe or taking an inordinate amount of computer time.

In its role as a prototype, the system is not designed to provide an exhaustive exploration of possible outcomes even within the PCS cost area. Rather, the model demonstrates the feasibility of using a control module, various data extraction and pre-processing modules, and a set of programs which permit variable formulation of a network assignment problem, to provide a decision maker with the ability to evaluate the staffing levels in varying scenarios.

The model uses a set of eligibility rules which may be changed by the user to test the results of matching the present population or "inventory" of the Marine Corps to a set of authorized billets under varying conditions. It is important to clearly distinguish the intended application of the model from two other related manpower functions: assignment and allocation.

Assignment, used in the manpower sense, is the precise matching of individuals to specific jobs. It is not possible in a mathematical model to capture all of the factors used in the assignment of officers. Many important but unquantifiable criteria, such as career patterns and past performance, must be considered by the monitors in the actual assignment of an officer to a particular job. Because of the size of the assignment problem and the complexity of the interactions involved, no single system can incorporate all of the factors which must be considered by the monitor. Therefore it is not the purpose of this model to make assignments of officers nor to mimic the assignment process.

Manpower allocation is the determination of those billets which are to be filled, based upon available manpower assets. Because it does not involve the actual matching of individuals to billets, allocation is a somewhat less difficult problem. The Marine Corps presently uses a system called the Officer Staffing Goal Model (OSGM) to set targets for the filling of officer billets. Functionally, the model described in this thesis bears many similarities to the OSGM. Both involve network optimization models which consider exactly the same substitution rules for matching people to jobs, but the decision support system described here is more flexible in accessing the data bases and permitting modification of some of the eligibility and matching criteria. In exchange for this increased flexibility, this model contains some simplifications of the rules used in the OSGM. For example, it does not include the capability to vary the share proportions for shortages within priority levels.

Despite the similarities, the model presented here is not designed to replace the OSGM. Rather, the model aggregates the results of the network problem to give the policy maker the ability to explore the effects of changes in funding level and policy. Thus, the model is intended for use as a decision support tool, *not* as an actual assignment or allocation system.

C. SYSTEM DESIGN

1. Overview

The system permits exploration of the feasible space of a problem with three principle dimensions: fill, fit and PCS cost. Several additional objectives, maximizing the fill in each of five job priority levels and sharing shortages within those levels, are incorporated in the model to reflect the guidelines that are observed in actual allocation. However, since these last criteria do not change in practice, they are not controllable by the user in the model.

2. Functional Organization

The system operates iteratively through six functional phases: data extraction, data preparation, formulation of the network problem, solution of the problem, summary of the solution, and control of problem re-definition and re-formulation through changing of policy data or weights.

In the first two steps of system operation, a series of modular programs are used to extract data pertinent to the allocation problem and to process it in preparation for input into a specialized problem generation routine. These programs update the input files, attach the matching rules to both people and jobs, and determine possible acceptable matches between individuals and billets. After this, the problem is formulated as a capacitated transshipment network in a program which attaches costs and bounds that may either be left at default values or adjusted by the user. Next, the formulation is solved using the generalized network solver, GNET [Ref. 1] which is presently available on the Marine Corps mainframe at Quantico, Virginia. After reviewing a summary of the solution, the decision maker then has the option of adjusting certain parameters to control the fill, fit, and PCS cost achieved in the solution. Alternatively, he may change the rules used in determining eligibility for transfer. In the prototype, the latter option is limited to raising or lowering the time-on-station requirements for reassignment in all billets, however the model could be expanded to include modifying the matching rules used to generate acceptable arcs, changing the size or organizational structure of the Marine Corps and many other factors which influence staffing levels. Subsequently, the modified problem is formatted and solved. The process may continue as long as the decision maker desires.

D. THESIS OUTLINE

This thesis presents the prototype for a system which utilizes current Marine Corps data on personnel, authorized billets, and substitution criteria to formulate a multiobjective optimization model tailored for use as a decision support tool. Chapter II, begins with a presentation of some background on the Marine Corps allocation and assignment process which will help in understanding the system. Next, the system is broken down into its six functional areas, and a brief summary is given of the programs used in each area. Chapter III deals with the development and formulation of the multiobjective optimization problem. In Chapter IV, conclusions and recommended areas for future enhancements are presented. Finally, listings of all of the source code and documentation are included as appendices.

II. DESCRIPTION OF SYSTEM

In order to permit the decision maker to analyze the effect of changes in policies or funding levels on staffing goals, it is necessary to simulate the allocation process which would be performed under those constraints. The prototype manpower decision support system presented in this thesis consists of a series of computer programs which prepare and solve the allocation problem in a way that bears many functional similarities to the present allocation system, the Officer Staffing Goal Model (OSGM.)[1] In addition, however, the system also provides the capability to repeatedly modify and solve the allocation problem. The system's operations may be grouped into six broad functional areas: data extraction, modification and preparation of the input files, formulation of the network problem, solution of the problem, presentation of the solution, and control of the re-formulation of the problem. Within each of these functional areas are programs, written in VS FORTRAN or SAS, which perform specific tasks related to that function. Overall control of these programs is handled by a CMS EXEC file which ensures that the programs are run in the proper order, and that the correct program calls are made after the user makes changes in the problem. In this chapter, each of the functional areas is examined, and the tasks and programs contained in each one are described. First, however, it will be helpful to explain some of the terms which will be used, and to present an outline of the Marine Corps allocation process as implemented in the OSGM.

A. TERMINOLOGY AND BACKGROUND

1. Explanation of Terms

Specific military skills within the Marine Corps are classified by a four digit number which codes the Military Occupation Specialty (MOS) of each individual. Every Marine is assigned a Primary MOS (PMOS) which indicates his or her particular area of expertise (eg. F-18A pilot, artillery officer, etc.) In addition to a PMOS, many Marines have one or more Additional MOSs (AMOSs) which they received as a result of schooling or demonstrated proficiency in some MOS. In some cases, the AMOS may be the same as the PMOS which is required for some billets. Other AMOSs refer to technical skills which may only be held as additional MOSs, such as Aviation Safety

[1]For a more detailed description of the OSGM than provided in this thesis, see the *OSGM User's Guide*, [Ref. 2.]

Officer (7596) or Operations Analyst (9650). MOSs may be grouped into more general categories called "occupational fields" (OCC fields) which contain all MOSs with the same first two digits. In the OSGM the OCC fields are even further aggregated into what are called "officer types." There are nine officer type categories in the OSGM, which include such broad groupings of OCC fields as "ground combat officer", "air-ground combat service officer", and "fixed wing pilot."

Warrant Officers and certain other officers have PMOSs that restrict them to certain billet types, usually of a very technical nature, and exclude them from certain command and staff billets. They are called Limited Duty Officers (LDOs) in contrast to "unrestricted officers" who may fill a wider range of jobs. Billets specifically requiring or excluding LDOs are said to have a "duty restriction."

The required MOS for a particular billet is called the Billet MOS (BMOS). In most cases, the BMOS corresponds to a PMOS or AMOS, but there are three BMOSs which do not have a PMOS or AMOS counterpart. These three are used for billets which have no specific PMOS requirement, but may be filled by any officer of a certain type. BMOS 9912 applies to billets that can be filled by any aviation officer. 9911 BMOS jobs may be filled by all unrestricted ground officers, and 9910 billets are open to any unrestricted officer, air or ground.

The individual who fills a particular billet need not necessarily have the same PMOS as the BMOS for that billet. In some cases, if an individual's AMOS is the same as the BMOS, he may be a candidate to fill that billet when no one with the correct PMOS is available. The set of substitution rules which define exactly who is qualified to fill a particular billet are contained in a file called the "Dictionary."

All commands at which Marines may be stationed are identified by a three character label called the Monitored Command Code (MCC). The structure of the Marine Corps is broken down into MCCs by grade (rank) and PMOS in the Authorized Strength Report (ASR). The ASR takes the total number of officers authorized by Congress, and distributes them among the MCCs by MOS and grade. The ASR is updated twice a year to reflect any changes in the structure or priorities of the Marine Corps.

2. Overview of the Allocation Process

The staffing allocation process involves distributing the limited manpower resources of the Marine Corps among the authorized billets so that certain objectives are met as closely as possible. In the OSGM, the objectives are solved "preemptively."

That is, the objectives are ranked in order of importance, and the highest priority objective is solved first. Subsequently, each of the remaining objectives is solved in order of its importance, using the solution to the previous problem as a constraint. This continues until either all of the objectives have been solved, or until there is no longer any flexibility left to improve the next objective.

The objectives of the OSGM are, in order of priority:

1. Maximize the total number of billets in the Marine Corps that are filled by qualified individuals.
2. Successively maximize the number of jobs filled in high priority billets without reducing the overall fill. The priority of a billet is defined by its Staffing Precedence Level (SPL) which is based on Marine Corps directives that set job priority policies according to the present need.
3. Minimize the difference in the proportion of billets filled within the same precedence level. This equates to sharing any shortages within SPLs.
4. Obtain the best "fit" for each person-to-job match. The only established quantitative measure of fit that has gained any degree of acceptance in the Marine Corps is the one used by the Officer Staffing Goal Model which defines up to five acceptable levels of substitution for each job. Each of these "fit levels" lists a set of eligibility criteria which must be met in order for a person to be matched to that billet. The criteria are established by the monitors and include grade (rank), PMOS, AMOSs, duty restriction, sex, officer type, and "experience." Because "experience" is difficult to define, it is seldom used by the monitors as a discriminator. Since the fit levels list the substitutions in the order of their desirability, obtaining the best fit amounts to minimizing the sum of all the fit levels.

In addition to these objectives which are considered in the OSGM, there is one more goal that the Marine Corps would like to include in the allocation process: minimization of Permanent Change of Station costs. The OSGM altogether lacks the capability of including these relocation costs among its criteria. Therefore, most of the recent PCS cost reduction has been done through either case-by-case decisions by the monitors or through broad policy initiatives, rather than by adjustment of the staffing goal. In an effort to cut PCS expenses, the Marine Corps is attempting to "reassign", rather than transfer, whenever possible. A reassignment is defined as any move that is less than 50 miles, or is confined to certain regions, defined by Marine Corps Order, where several Marine installations are close by. When a Marine is reassigned, he is expected to continue at his present residence; hence no relocation costs are incurred by the Marine Corps.

B. FUNCTIONAL ORGANIZATION OF THE MODEL

In this section the system is described through a detailed presentation of the six functional areas. An overview at the beginning of each discussion lists the tasks that are performed in that area with an emphasis on any special design considerations that affect more than one program. Next, the implementation of the tasks is described in detail by looking at how the individual component programs contribute to the accomplishment of the tasks. When the design of an individual program is particularly complex, a list of the tasks in the program is also presented.

1. Data Extraction

a. Overview and Task Listing

Since data extraction is dependent on parameters that the user does not input until *after* the first solution is achieved, it is necessary to first initialize certain files that will be used to set eligibility requirements, weights, and objective function priorities before the problem can be solved. After this, the supply, and demand are extracted, along with the rules used to update and match them. Pertinent information on the inventory of officers (supply) is drawn from a file called the Headquarters Master File Extract or HMF Extract (in the prototype, it is called "WORKING INVENTRY") which contains personal information on all Marines and is updated bi-weekly. The demand for billets is read directly from a file containing the ASR. . The information used to update and match the inventory and the ASR is contained in a series of files that are collectively known as the "Dictionary." Since the prototype deals only with aviation-related billets, only those officers, jobs, and Dictionary records applying to them were used. The files which were created for this reduced organization have the same format as the complete Marine Corps files and were taken directly from the actual unprocessed data that would be used in the full scale model.

The data extraction tasks, which are explained in detail in the next section, may be summarized as follows:

1. Initialize the files used to define the problem.
2. Extract the inventory of officers from Marine Corps data files. This includes reading in an adjustment factor to the normal tour length from a user-controlled file, re-coding certain character variables into integers as they are read in to speed up data preparation, attaching a cost location code to each record, and formatting and sorting the files for the data preparation phase.
3. Read in the demand file.

b. Programs

This section contains a description of the programs used to effect the data extraction tasks.

(1) *File Initialization (Program name: INITLIZE Language: FORTRAN).* Since the prototype initially solves the problem using current policies and priorities *before* the user interacts, all of the files which will contain user-supplied parameters after the first run through the solver must be initialized so that the problem can be solved the first time. After the initial solution is achieved, the user is given control over certain parameters relating to the extraction of the inventory and the formulation of the problem. The values he selects are written to files which are, in turn, read by the inventory extraction and problem generation programs on later runs. Thus, INITLIZE is bypassed on all subsequent iterations of the formulation and solution process.

(2) *Inventory Extraction (Program name: FREE-FIX Language: SAS).* The supply for the allocation problem is generated by separating the officer inventory into two groups, those who are eligible to be transferred, called "movers" or "free" officers, and those who are not, called "non-movers" or "fixed" officers. Free officers are those who have been at their present duty at least as long as the "standard" tour length at that billet, which is defined by the Tour Control Factor (TCF) for the billet. The TCF is the length, in months, of the "standard" tour length for a particular billet, and varies depending on the geographic location, type of duty station, and whether the individual is accompanied by his family or not. Movers may be transferred to any billet for which they fill at least one of the acceptable substitutions found in the Dictionary. Non-movers may only be reassigned to billets to which they match within their current MCCs. Determination of who are movers and who are non-movers is based upon those whose standard tour is over before the end of the period for which the allocation is to be run. This period, which is usually one year, is called the "window" of the allocation problem.

In the prototype, the user may adjust the TCF to reflect changes in tour length policies. This tour length adjustment factor is read into FREE-FIX using a SAS macro. In the first formulation, the factor is set to zero, meaning that the normal tour length policies hold. At the end of each solution summary, the user is permitted to change the TCFs through adjustment of this factor. After it is read into FREE-FIX, the adjustment is applied equally to all billets to determine new TCF's used in extracting a new set of free and fixed officers for the next problem.

In addition to dividing the officers into movers and non-movers, the inventory extraction program converts many of the data items from character to integer values as they are read in. This permits the use of 0,1 or 0,1,2 rather than character comparisons in the matching routines, and allows declaration of smaller storage space if desired. Also, within the inventory extraction module, a "Cost Code Center Index" (CCCI) is assigned to the duty station of each officer which lab^els his general geographic area. Cost Code Centers are locations used to aggregate the nearly 2400 MCCs in the Marine Corps into 63 geographic areas. The CCCI is used later in estimating the relocation cost. Finally, the fixed and free inventories are sorted two ways and output to appropriate files. The first pair of files (named USMC MOVRSUP and USMC NONMSUP, respectively) contain the free and fixed officers sorted by MOS, grade, and MCC. The second two files (MOVERS SORTXOTYP and NON-MOVR SORTXOTYP) are sorted by officer type and grade. Both sets of files are designed to be read into the matching program, however only the first two are processed further in the prototype. This is because only the MOS/grade criteria substitution matches were considered in the prototype matching routine.[2] Output from FREE-FIX includes files of fixed and free officers appropriately sorted for the matching programs, and a file containing data on those who are expected to leave the Marine Corps because of retirement or the end of their obligated service.

The tasks performed by FREE-FIX can be summarized in the following list:

1. Read tour length adjustment factor from file TCF-ADJ DATA.
2. Extract supply of fixed and free officers from the HMF Extract based on the most recently calculated values of the TCFs.
3. Remove from this preliminary inventory those who will be retiring or getting out due to the end of their obligated service.
4. Recode character variables into integer where possible, and reformat files for subsequent use.
5. Sort and output the free and fixed inventories to files to be used later.

[2]. ·re are three basic ways of sorting and classifying the substitution criteria: MOS and grade, AMOS or OCC field and grade, and officer type and grade. Only the criteria involving the PMOS and grade were used in the prototype to simplify the problem. See the discussion on the need to generate all matches in Chapter IV, Section B.1.b.(1).

(3) *Demand Extraction* (*Program name:XPASR Language:SAS*). The demand is defined by the Marine Corps in the Authorized Strength Report (ASR) which contains a list of the number of authorized billets sorted by MCC, MOS, and grade. The ASR is read in and reformatted for the data preparation phase.

(4) *Dictionary Extraction.* Many of the files needed for the solution of the allocation problem have already been developed for the OSGM Dictionary. Because of the need for quantifiable measures of "fill" and "fit" and the difficulty in establishing acceptable rules for people-to-billet matchings, the rules used in this model are adopted directly from the Dictionary. This eliminates the necessity for establishing a separate standard of measure which would have to be validated and maintained by the Marine Corps. The files in the Dictionary set priorities for the filling of billets, define acceptable substitutions for each job, define and rank "fit" for each acceptable substitution, establish a matching from the substitution list to the demand, and permit changing the numbers and types of both supply and demand. In the prototype, the Dictionary files are read in as needed. There is therefore no program dedicated to this task.

2. Data Preparation

Data preparation involves updating the demand requirement and generating potential matches of people to billets. This is accomplished through the use of a demand adjustment file, pointer arrays, and by applying the matching rules contained in the Dictionary separately to the fixed and free inventories.

a. Overview and Task Listing

(1) *Overview.* Matching people to jobs is potentially the most time consuming part of the data preparation phase since there could be several hundred jobs that each of the several thousand (approximately 7,000 in the prototype, and 17,000 in the full scale model) people in the inventory could fill. Whether the actual search is conducted by searching through the inventory for each billet or by searching through the billets for each person, it involves a very large number of criteria comparisons since each substitution record has nine items that define the substitution.

There are several possible approaches to accelerating the task. One method would entail simply deleting all of the non-movers and their billets from consideration. But this would eliminate the possibility of reassignment of non-movers within their current MCCs which might substantially reduce PCS cost.[3] Another

[3]In the prototype, reassignments are limited to the MCC in which the individual is currently located. In practice, however, they could be made to any other MCC

approach involves the total enumeration of all possible matches between individuals and billets, followed by elimination of matches involving transfer of fixed people out of their MCCs. But this technique is computationally wasteful and requires more computer time.

The approach taken in the prototype lies between the two listed above. In this method, movers and non-movers are matched separately through the use of an additional, smaller ASR composed of all billets not already filled by fixed personnel. This reduced demand list, FREE ASR , is used in the matching of movers, while non-movers are matched to demands within their MCC that are in the complete demand list, WORK ASR. The combined list of potential matches that results includes almost all possible reassignments yet involves a much smaller number of criteria comparisons and avoids generating unnecessary arcs to billets where there is no demand. In exchange for the reduction in the size of the matching problem, the list does *not* include matches of free individuals into fixed billets in those cases where a fixed person is reassigned within his MCC. This should not dramatically affect the overall solution since many MCCs have only a single demand, and where there are multiple demands, most individuals can be matched to only one or two billet types. Thus, in cases where there is more than one job for an individual within the same command, the impact (if any) on the solution will usually be limited to a slight change in the fit solution. This sacrifice was deemed acceptable in order to achieve a large reduction in the number of comparisons that have to be made by the matching routine.

The model employs numerous indices to refer to data items which can be grouped together. This permits reference to a single number rather than to multiple data elements all related to the same object. For example, individuals are referenced by an index called "IDNUM" rather than by all of the data relating to each person. Similarly, each unique billet requirement is referred to by means of a nine character name called the Billet Officer Description (BOD). Each BOD is defined by substitution rules that may contain up to eleven criteria.[4] Rather than carry all of the eleven criteria along until they are used to match people to billets, or even carrying a nine character BOD, the BOD is identified by a four digit integer BOD number (BODNUM) which

within a 50 mile radius of the current MCC, or, in the case of those which fall into certain regions defined by Marine Corps directive, to any MCC within their region. See Chapter IV, Section B.1.b.(7) for a discussion of how this might be implemented.

[4]lowest acceptable grade, low grade experience, highest acceptable grade, high experience, PMOS, officer type, first AMOS, second AMOS, sex, duty restriction, and fit level.

serves as an index to a file where all of the criteria for each BOD are stored until they are needed. Also, individual billets are referenced by billet numbers (BILNUMs.)

Potential people-to-job matches are found by first identifying all individuals (indexed by IDNUM) who fill the substitution criteria for a certain job type or BOD. Next, all billets requiring that BOD are identified and indexed by a BODNUM. Finally, the index number (IDNUM) of the individuals who are matched to a BOD are matched to the index of the specific billet (BILNUM) by merging according to the shared BODNUM. This heavy use of indices complicates the reading of the programs somewhat, but it reduces the amount of data that must be carried from one file to the next.

(2) *Task Listing.* The data preparation phase begins with the generation of pointer arrays which are used later in the matching program. The pointer arrays give the indices of the first and last occurrence of a particular MOS/grade combination in the sorted inventory lists. Next, the ASR is updated to incorporate any recent changes to the demand. Third, the fixed inventory file is re-formatted and input to a program where billets occupied by non-movers are subtracted from the complete, updated ASR to give a smaller ASR which is used in the matching of the movers. Following this, both the complete and the reduced ASRs are broken down into more specific billets with more detailed job requirements, and a BOD is attached to each billet. Using the BOD as an index, a set of acceptable substitutions are then attached to the billet. Next, the entire inventory is matched to the billets according to whether the individual is fixed or free, using the substitution lists that were attached to the demands. Finally, the resulting list of potential people-to-job allocations is sorted for input into the network generation program.

The list of tasks for the data preparation phase is summarized below:

1. Generate pointer arrays to the inventory Files.
2. Update the ASR file to reflect changes in the structure of the Marine Corps. (Resulting file: WORK ASR)
3. Re-format non-movers for subtraction from the full ASR.
4. Generate a reduced ASR file which does *not* contain any billets that are already occupied by non-movers. (Resulting file: FREE ASR)
5. Split both WORK and FREE ASR files up into more detailed demands, and join those detailed billets to their corresponding BODs. Also, attach the substitution lists to the billets using BODNUMs to link the two files.
6. Match the inventory to the billets using the substitution lists to link the two files. This generates a list of potential people-to-job matches.

7. Sort and format the list of potential allocations for input into network generation program.

b. Programs

In this section, the programs which implement these data preparation tasks are described.

(1) *Generation of Arrays (Program names: ADJ-LIST, ENTRY-PT Language: FORTRAN).* The first task accomplished in the data preparation phase is generation of pointer arrays. In order to speed up the process which matches the inventory to the demand, an array is generated for each of the sorted inventory files, which points to the first and last occurrence of a particular MOS and grade in both the fixed and free inventory files. This greatly reduces the number of comparisons that must be made in the sorting program by limiting the search for matches to the precise MOS and grade desired.[5]

(2) *ASR Update (Program name: C1ASR Language: SAS).* Following generation of the pointer arrays, the ASR is updated to reflect changes in the numbers of people authorized at various billets which have occurred since the ASR was last revised. The file containing these changes, WKC1 CRD, may also be used to add or delete entire units or billet types to or from the Marine Corps. The resulting file, WORK ASR, thus reflects the structure of the Marine Corps which the decision maker wishes to use in the analysis of staffing policies, priorities, and constraints.

(3) *Re-Format Non-Movers (Program name: ROLNM Language: SAS).* This program converts the non-mover inventory up into ASR format in preparation for generation of the reduced ASR.

(4) *Generate Reduced ASR for Matching of Movers (Program names: INVNTRY1, ADJ-LIST, EXCESS Languages: SAS, SAS, FORTRAN).* In order to reduce the number of comparisons which must be made in the matching program, movers and non-movers are matched to their respective demands separately. This series of programs produces the special ASR that is needed to define the reduced demand for movers by subtracting from the complete ASR all billets that can be definitely identified as already occupied by fixed personnel. Not all billets occupied by non-movers can be linked to the ASR without looking at the set of substitutions, even

[5]A similar entry point array should be made for the inventory files that are sorted by officer type and grade in any full scale implementation of the system. In the prototype, however, this was not done since the matching was performed only on MOS grade substitution criteria. (See the discussion of the Matching Routine below in Section B.2.b.(6).)

though this is precisely what the programs seek to avoid. In lieu of time-consuming sorting through all substitution list sets, these programs use a set of conservative matching heuristics to quickly identify and eliminate from the ASR most of the billets occupied by non-movers. In doing so, they provide a much smaller list of billet substitutions which must be searched to determine the potential matches for movers, and reduce the number of unnecessary arcs generated in the network.

For example, suppose that there is an aviator who is fixed at a command which does not have any requirement for his specific PMOS. The heuristic will seek to determine the billet he is filling by looking for a 9912 BMOS billet (which can be filled by any aviator) and any billets for which his AMOS might qualify him. If it can match him to an authorized billet, his billet can be eliminated from consideration, thus reducing the number of matches that must be made.

(5) *Split ASR into Precise Demands and Attach Substitution Lists to Billets* (*Program names*: *E2ASRA*, *E2ASRB*, *E1ASRE2A*, *E1ASRE2B*. *Language*: SAS). Before individuals can be matched to jobs, the specific job substitution criteria given in the Dictionary, which contain nine matching categories,[6] must somehow be matched to the broad authorization categories listed in the ASR which are identified only by PMOS and grade. This is done in two steps. First, the MOS/grade combinations for each MCC given in the WORK ASR and FREE ASR files are broken down into specific billet requirements and linked to a particular job description, identified by the Billet Officer Description (BOD). Next, a set of up to five acceptable substitution criteria is attached to each specified billet using the BOD.

For example, suppose that the ASR lists a demand for twelve Captains with MOS 7564 (CH-53D pilot) at a particular helicopter squadron. The actual billet need in that unit may include one specially trained Aviation Safety Officer (MOS 7596), while the rest of the jobs may only require a 7564 Captain (squadron pilot.) Therefore, the demand for twelve 7564 Captains must be broken down into a demand for one BOD called "Aviation Safety Officer" and eleven with a BOD of "squadron pilot". Both BODs have a different set of substitution criteria in the Dictionary.

The mechanics of attaching the list of acceptable substitutions for each specific job to the ASR are not complicated. First, the ASR is split into detailed billet requirements through rules contained in what are called "F2 Cards."[7] The E2 Cards

[6]PMOS, first AMOS, second AMOS, lowest acceptable grade, low grade experience, highest acceptable grade, high grade experience, sex, and duty restriction.

[7]The word "Cards" on many of the OSGM files is a carry over from the early

break apart the ASR demand at each MCC into specific billets and attach both a BOD and a Staffing Precedence Level (SPL) to each job. Substitution lists are then attached to the specific demands that were identified by the E2 Cards using what are called "E1 Cards." The E1 Cards define an ordered set of acceptable substitutions for each BOD. Thus, the BOD is the essential link between the ASR demand and the substitution lists.

The first substitution listed for each BOD is given a fit level of one, the second, a fit level of two, and so forth. Many BODs have only one acceptable substitution description, but there may be up to five listed per BOD. Each substitution does not have to be completely explicit, but may indicate an acceptable range of choices or even complete indifference about certain of the criteria. For example, a substitution may permit any PMOS within a certain OCC field and may express indifference about the duty restriction criteria. The BODs are attached to the free and fixed ASR demands by the programs E2ASRA and E2ASRB, respectively. These, in turn are merged with the substitution lists in E1ASRE2A and E1ASRE2B, respectively. The latter two programs also append the appropriate Cost Center Code to each MCC in the demand list.

(6) *Matching Routine (Program Name: MATCH-AL Language: FORTRAN).* Once all billets have been explicitly identified and defined and their substitution lists established, the next task is the actual matching of the fixed and free inventories to the specific jobs. In the prototype, matching is accomplished by finding all individuals who match to each BOD, then finding all jobs associated with that BOD. The people-to-billet matches which result from the merging of that information are then written to a file called RAW ARCS, since the potential matches they represent constitute the initial set of arcs used in the network formulation.

MATCH-AL program operation is performed first on the movers, then on the non-movers. The functional tasks accomplished in MATCH-AL are listed below:

1. Read a 63 × 63 MCC to MCC cost matrix into resident memory. This is used to assign a PCS movement cost to all legitimate matches found in the program and is only used when matching movers since reassignments have no PCS cost. The costs used in the prototype are scaled to be representative of actual costs, but are not based on detailed cost studies. Rather, they are equal to a constant (2,000) added to the distance between the MCCs that are being matched.

days of the OSGM when actual punch cards were used.

2. Read the complete free (or fixed, in the second part of the program) inventory into resident memory. Each individual characteristic that appears in the substitution cards is read in array form for every person that appears in the inventory. In addition to the seven variables[8] required for the substitution criteria items, five more indices[9] are used to access some cost and pointer arrays, to index the individual and his location, and to ensure that no one is matched to the same billet more than once. In practice the 20000×12 array which was used in the prototype is larger than would ever be needed in a full scale model, even if everyone were treated as a mover. The purpose of this seemingly extravagant use of memory is to speed up the matching by eliminating costly accessing of the inventory files each time another substitution card is read. Careful count is kept of the number in each inventory file so that the same array can be used for both the fixed and the free matches. After all movers have been matched, the array is over-written with the array of non-movers.

3. Read in the pointer array for the inventory file that is being matched. When searching for those who might meet the criteria for a a particular substitution, the pointer array is used to go directly to the MOS and grade listed in that substitution criteria, thus avoiding a lengthy search through the entire inventory.

4. Read in each "demand group" from the E1ASRE2A and E1ASRE2B files. A "demand group" is defined as the set containing all of the substitution cards (ie. E1 Cards) and demand for a particular BOD (ie. all MCCs at which that BOD requirement is found, and the number of empty billets at each MCC.)

5. Find all individuals in the inventory who match the criteria listed in every substitution card. This is done as follows: First, for each substitution card, using the pointer array to narrow the search, find every individual or group of individuals who meet the criteria and have not previously been matched within that BOD. Second, generate arcs (matches) between that individual or group and all MCCs in the demand group. Third, mark the individual or group to avoid re-matching within the same BOD at a lower fit level. Lastly, continue the search until there are no more possible matches for that substitution, and then go to the next card. Entire groups may be matched when the only substitution criteria is grade and MOS. This occurs on almost fifty percent of the E1 Cards, and makes it possible to directly match individuals to MCCs by using the pointer array as an index to the IDNUMs (individual indices).

[8]PMOS, grade, first AMOS, second AMOS, sex, duty restriction, and experience.

[9]The five indices and their uses are:

ICOSTC	- Individual Cost Center Code index used to enter cost array.
IDNUM	- Identification index for use in analysis of the solution.
IMOSNO	- MOS index number used to access pointer arrays.
LSTBOD	- The last BODNUM to which the individual was matched. It is used to prevent multiple arcs from a person to the same BOD
PMCC	- Present MCC. Used to determine if reassignment is allowed.

The result of performing this process for both fixed and free inventories is a file containing the legitimate people-to-billet matches. Although the fixed inventory search covers the entire ASR demand, by limiting eligible matches to the same MCC, the number of comparisons that must be made is greatly reduced.

There are three primary ways of matching individuals to billets based on the type of E1 Card used for a given billet. Approximately seventy percent of the E1 Cards contain a specific PMOS and six other criteria: grade, AMOS1, AMOS2, experience, sex, and duty restriction. Less than ten percent of the E1 cards indicate indifference about the PMOS, and are primarily concerned with AMOS1 and the remaining five criteria. The remaining twenty percent of the E1 Cards have an officer type in place of the PMOS and have the same six other criteria as the first type of substitution. Each one of these E1 Card types requires over 500 lines of FORTRAN code to exhaustively identify for both inventories. Since the purpose of this thesis is to demonstrate the concept of the overall system, and not to construct the full scale model, the prototype performs matches only on the first type of E1 Card. However, the program is written so that appropriate subroutines could be inserted to incorporate matching on the other two types of cards without disturbing the flow of the program. Notwithstanding the present limitation, the output from the model does not appear to be unreasonable. (See discussion of output presentation in Chapter IV, Section B.1.b.(7).)

(7) *Sorting of Matching Routine Output* (*Program name*: *BIGSORT Language*: *SAS*). This program sorts the output from MATCH-AL by SPL, BOD, and MCC in preparation for input into the network generation program.

c. Summary of Data Preparation

The programs in the data preparation phase combine the supply and demand of the allocation problem in such a way that the network problem can be formulated directly from user inputs and a single data file containing all basic people-to-job matches.

3. Generation of the Network Problem (Program name: NET-GENX Language: FORTRAN)

Generation of the network problem consists of several tasks:

1. Determine all legitimate arcs in the capacitated transshipment problem network.
2. Calculate costs on those arcs.
3. Format the problem for input to the solver, GNETBX.

This program converts the sorted people-to-billet matches into the capacitated transshipment network described in detail in Chapter III. It allows for the changing of weights or priorities attached to the fit and cost objectives, and the adjustment of a lower acceptable bound on the total number of billets filled. This flexibility is achieved by the use of parameter input files which are used to determine the values of certain weights and bounds in the formulation. During the first formulation of the allocation problem, the program uses the parameters that were set by the program INITLIZE. On subsequent iterations through the system, the user may adjust these parameters which will then change the weights and bounds in the formulation. Refer to Chapter III for additional details of the program and how it adjusts the formulation of the model to reflect the user's desires regarding priorities and goals.

4. Solution of the Network Problem (Program name: GNETBX Language: FORTRAN)

The capacitated transshipment network formulated to capture the allocation problem is solved using GNETBX, a software package developed by Bradley et. al. in 1975. A detailed explanation of the operation of the solver can be found in 1.

5. Presentation of the Solution (Program name: SUMMARY Language: FORTRAN)

This program presents some brief statistical summaries of the solution to the user. In the prototype, the summaries provide only the most general overview of the solution. In a full scale implementation of the system , the user will want the option of exploring the distribution of resources in the allocation solution in much greater detail.

6. Problem Re-formulation (Program name: SUMMARY, THESIS Language: FORTRAN, CMS EXEC)

The process of re-definition of the problem is handled by a control module which gives the decision maker the opportunity to modify the formulation by changing the data extraction rules (ie. the TCF), the fill bound, or the weight placed on the fit and cost objectives. The user has the option of changing one or all of the parameters and may reset them as many times as he desires before leaving the control module. Once he exits the FORTRAN control program, control reverts back to the VS EXEC program, THESIS, which ensures that all of the programs are executed in accordance with user instructions.

If the user has changed the TCF adjustment factor and indicates that he wishes to see the resulting new optimal allocation, the EXEC program loops back to the beginning of the extraction process, bypassing INITLIZE and re-entering FREE-FIX. This causes all subsequent programs to be executed again in order. If he has changed the objective priorities or the limits on fill and desires to re-calculate the allocation, the EXEC program re-starts the solution process at NET-GENX. Since no change has been made in the inventory or demand, only the network with its associated weights and bounds need be re-formulated and solved.

In a full scale implementation there are many other places the system could be directed to re-start. If the decision maker were given the option of changing the substitution or matching rules, the process should re-start before the substitution cards are attached to the billets (ie. E1ASRE2A and E1ASRE2B.) If the user were to change the structure of the Marine Corps by modifying the ASR adjustment file (WKC1 CRD) to add or delete units or billet types, the process should be re-entered where the ASR is updated to reflect changes which have occurred since the last semi-annual update (ie. C1ASR SAS.) The sequential and modular structure of the system thus permits enhancements without necessitating any changes in the basic program design or organization. These possible enhancements are discussed more fully in Chapter IV.

III. DESCRIPTION OF THE NETWORK ASSIGNMENT MODEL

The heart of the prototype system presented in this thesis is a specialized network model which may be modified by the user. This chapter begins with a discussion of the multiobjective optimization techniques that are used in the network model.[10] Following that, the model used to solve the allocation problem is described and the mathematical formulation is presented and discussed.

A. MULTIOBJECTIVE OPTIMIZATION METHODS USED IN THE MODEL

Multiobjective optimization is defined by Rosenthal in [Ref. 3] as the field of optimization dealing with problems that have more than one measure of effectiveness and a feasible region that is too large to enumerate. Let F(x) be a multiobjective function composed of n component objective functions $f_1(x), f_2(x), \ldots, f_n(x)$. Since there is no clear way of ordering (and, hence, maximizing or minimizing) a vector valued function such as F(x), various approaches have been proposed to deal with this class of problems. Although there is no universally acknowledged classification of these techniques, Rosenthal [Ref. 3] gives three basic categories of approaches which apply to the model presented in this thesis: Priorities, Aspiration Levels, and Weights. All of them involve some degree of subjectivity and each has some significant weaknesses. A brief discussion of each one and how it is used in the model will assist in understanding the network formulation, which combines all three.

1. Priorities

The Priorities approach, also called Preemptive or Lexicographic Programming, involves solving an ordered set of component objective functions, $f_i(x)$. In one implementation of the approach, each objective is sequentially optimized in order of its priority. If there are any ties in the solution, then the next ordered

[10]The techniques discussed in this chapter apply to the solution of the problem once it has been formulated, and do not deal with the assumptions and heuristics that are used to develop that formulation. For example, in the prototype, only those E1 cards involving MOS and grade criteria were used, while those substitutions based on officer type, OCC field, or AMOS were ignored in order to reduce the size of the problem. Therefore, the solution returned from the solver would be optimal with respect to the formulated problem, but the simplification used to make the formulation would render the solution sub-optimal with respect to the actual overall allocation problem.

objective may be solved. This continues until either all component objective functions are solved, or until there are no more ties and, thus, there is no flexibility remaining for improving the lower priority objectives. Although this preemptive method lends itself readily to mathematical programming techniques, it requires a large number of ties if there are many objectives and does not permit tradeoffs among the objectives which often occur in practice.

It is also possible to solve the multiobjective optimization problem preemptively in a single pass if sufficiently large weights are used to separate the objectives. This is the technique employed in the model. One potential danger in this approach is the requirement for large weights which could cause roundoff error if they are not chosen properly.

The first priority in Marine Corps allocation is maximizing the total number of billets filled. Operationally, this objective is treated as paramount, therefore the fill objective is solved in an essentially preemptive fashion.

2. Aspiration Levels

This approach consists of setting targets or goals for one or more of the objectives, then optimizing the rest. Aspiration levels can be expressed as either ideals or as minimum acceptable values, b_i. Since we cannot speak of either minimizing or maximizing F(x), the problem can be stated: "Find the 'most favorable' (however that may be defined) value of F(x) subject to the constraint,

$f_i(x) \geq b_i$ (for all i with aspiration levels.)"

In the formulation of the model, a lower bound may be placed on the acceptable level of fill. This permits the user to keep the preemptive ordering of the fill objective while allowing additional ties to improve the solution in the other criteria categories.

3. Weights

By multiplying each component objective function by an appropriate weight and then optimizing the sum of the weighted costs, the multiobjective optimization problem can be solved as a single criteria optimization problem. The resulting utility function,

$$U(x) = \sum_i w_i f_i(x) \quad \text{(for all i)},$$

can then be optimized over the range of x. This is the basic approach underlying the prototype model. However, it has been modified slightly to permit the use of aspiration levels, and to account for a number of inherent weaknesses in the weighting approach which are discussed below.

a. Potential Problems with Weights

(1) *Constant Marginal Returns.* Using constant weights on the objectives that are being considered ignores the principle of diminishing marginal returns. This principle states that an individual is more willing to tradeoff some amount of a commodity when there is a lot of it than when there is very little. Similarly, using a fixed weight to combine two objectives does not reflect the fact that, as one objective approaches an optimal value at the expense of the other, the extreme condition of the one may cause the decision maker to change his relative valuation of the two objectives. This deficiency can be partially overcome by allowing adjustment of the weights after each solution so that the decision maker can vary the importance of each objective according to the level each one has attained. Unfortunately, this may become confusing to the user if too much manipulation of weights is performed. In the prototype, the relative weights of only two objectives, fit maximization and PCS cost minimization, are controlled by the user. Additionally, control is limited, for the sake of clarity and modeling integrity, to the setting of one or the other as preemptive, or setting them to equal priority. Using pairwise combinations of related objectives greatly simplifies interpretation of the changes from one solution to the next.

(2) *Subjectivity and Implicit Assumptions.* Although mathematically simple, weights may be difficult to determine and validate operationally. Weights determine the relative emphasis on the objectives and the relative value of tradeoffs, yet they invariably involve the personal judgment of the designer, and are often buried within the computer code of a formulation. It is therefore important that the assumptions used to derive the relative weights be explicitly stated. It is even more preferable to give the user the ability to set or change the weights himself, provided this is done carefully.

(3) *Non-Compatibility.* Although different objectives can be combined into a single utility function using a weighting scheme, their measures may be incompatible. In the prototype, PCS cost is a ratio number, meaning that the ratios of PCS costs have meaning. For instance, a $6,000 PCS move is actually twice as expensive as a $3,000 one. But fit is measured on an ordinal scale, so differences in fit numbers indicate only relative ranking. A fit level of 4 is not necessarily twice as "bad" as a fit level of 2.

When ordinal and ratio numbers are combined into a utility function, it is important to avoid any invalid statistical manipulation of the resulting values or scores. In the prototype, this danger is reduced by conversion of the implicit utility values back into the un-scaled values of fit and cost in the solution. Thus, it is not possible for the user to compare them directly. Furthermore, the user is not given the option of fine-tuning the weights in order to prevent him from trying to achieve some precise desired solution solely by toying with the weights.

(4) *Required Large Size.* Another potential problem with weights is present when weights are used to enforce the preemptive solution of a large number of objectives. If there are many lexicographic objectives, the weights required may be so large that computer accuracy limits are exceeded. In the model described here, there are nine objectives in the formulation: maximize total fill, maximize fill within each of five priority levels, minimize the variation in proportionality of fill within each priority level, maximize "fit" as defined by the E1 Cards, and minimize PCS cost. Thus, even though the objectives are not all solved in a strict lexicographic order in the model, the size of the weights is a concern in the formulation.

Fortunately, it is not essential in a decision support system, such as the one described here, to have prefectly impermeable boundaries between the objectives. In practice, if a large gain in a secondary objective can be obtained by the sacrifice of a "small" amount of a primary objective, the tradeoff might be preferred. Thus, in order to keep the magnitude of the weighting coefficients manageable and to provide the opportunity for extremely "profitable" tradeoffs between some objectives, the differentials between objective function weights should be adjusted to capture the desired level of separation among the priorities.

b. *Measure of the Preemptiveness of Objectives When Using Weights*

When dealing with a large number of objectives, there is no simple and efficient way of determining the measure of preemptiveness of an objective relative to objectives with lower priorities. It is usually not practical to enforce the preemptive solution of multiple objectives simply by using large weights. Other knowledge of the system being modeled must be included in order to reduce the required size of the weights to within computer accuracy limits. By doing so, smaller weights may be used. This section deals with methods of evaluating whether the weight placed on an objective is sufficiently large to ensure that it is solved in the desired order.

The degree of separation between objectives can be roughly measured by the number of billets in the next lower priority objective which would have to improve from the least to the most preferred category within that objective before a reduction in the higher precedence objective would occur. For example, suppose there are three ordered objectives with priorities and costs as shown in Table 1.

TABLE 1
WEIGHTS OF THREE OBJECTIVES IN A MULTIOBJECTIVE FUNCTION

Objective	Priority	Weight Range or value
1	1	-100000 to -10000
2	2	-100 to -10
3	3	-3

Each of the weights on the arcs associated with the first two objectives lies within a range of values. The third objective arc has a single, constant weight. The number of improvements in the second objective function from the worst to the best case which must occur before there will be a single reduction in the the first objective function can be calculated by taking the difference between the highest (worst) cost of any objective one flow and the lowest (best) cost of any objective two flow, and dividing it by the maximum improvement that could occur in a single objective two flow. This gives N_1, the number of additional flows in objective two needed to balance a reduction in objective one.

$$N_1 = |(-100 - (-10000)) \div (-10 - (-100))| = 110$$

A slightly different measure can be used for objective three, which has a constant weight along its arc. The number of additional units of flow into objective three that must occur to equal the value of a single flow through objective two is :

$$N_2 = |(-3 - (-10)) \div -3| = 2.33$$

N_1 implies that any reduction in objective 1 by a single unit would have to be matched by at least 110 units of objective 2 improving from the worst to the best category to make them of equal value. N_2 implies that a reduction in objective 2 by a single unit would be acceptable if more than 2 more units could be brought into the solution in objective 3. It should be clear that nothing definitive about the preemptiveness of the objectives can be inferred from these numbers without also looking at the possible, or at least probable, number of units that may flow through the nodes of the network. If there are fewer than 110 units of flow possible into objective 2 or less than three into objective 3, then both objectives may be regarded as preemptive. If that is not the case, then objectives 1 and 2 cannot be considered strictly preemptive using the above weights. However, if evaluation of the matching or assignment rules indicates that no more than, say, five or ten units are affected by any single other unit with respect to objective 1, or one or two units with respect to objective 2, then both objectives may be considered to be "operationally preemptive." In the prototype, many of the objectives are "nested" or contained within higher priority objectives, thus the only requirement on those weights is that they be set to a value higher than the best possible value obtainable by achieving all lower priority objectives.

Determination of operational preemptiveness is difficult and may entail analysis of the distribution of assignment criteria, or even simulation. In fact, experimentation with the weights can and should be done regularly on a working model to verify that they are sufficient to maintain the desired degree of separation between objectives without carrying excessively large coefficients. As can be seen from the size of the weights of the first objective in the example, unnecessarily large weights can quickly reduce the number of objectives that may be considered before the size of the largest weight exceeds computer limitations.

Table 2 lists the nine objectives in order of their priorities along with their weights. Note that none of the weights is large enough to *guarantee* that no tradeoffs will occur since there is a very large flow through the network. Thus, none of the objectives is strictly preemptive. However, it can be seen by analysis of the substitution rules and experimentation with various weights, that several of the objectives act preemptively. Because of the nested nature of the SPL objectives in the network, the size of their weights are sufficient to ensure that the SPL's are filled in order, despite their closeness. Because each billet is associated with a single person (and thus, a single PCS cost and fit), there is no chance that a single assignment will affect more than one PCS cost or SPL fill proportion at a time, and therefore, there is no chance of overlap. Hence, the last three objectives are operationally preemptive.

TABLE 2

WEIGHTS ATTACHED TO PROTOTYPE OBJECTIVE FUNCTIONS

Objective	Priority	Range/coefficient	Symbol
Fill	1	-99999	a_{hs}
SPL 1 Fill	2	-40000	$e1_p$
SPL 2 Fill	3	-30000	$e2_p$
SPL 3 Fill	4	-20000	$e3_p$
SPL 4 Fill	5	-10000	$e4_p$
Fit	7	-7999 to -100	c_{ij}
PCS Cost	8	-99 to -10	c_{ij}
Proportionality of Fill Within SPL	9	-9 to 0	$w^k{}_{jh}$

Note: Symbols are explained in Section 3.1

4. **Priority of Objectives**

The first objective that must be solved is that of maximizing "fill", which is measured by the total number of billets filled in all categories. In practice this objective is regarded as most important, so in the model it is included as the first pre-emptive objective. Consequently, the arc along which the fill must flow in the network will normally have no upper bound. However, since the use of pre-emptive criteria precludes the very reasonable practice of allowing a slight tradeoff in a higher ordered objective for an extremely large gain in a lower one, the model allows the decision maker to reduce the fill to improve the other objectives. This is done by setting a maximum flow through the fill arc, which has the effect of setting the selected upper bound as an aspiration level. This relaxes the fill requirement and permits more flexibility in improving the fit and cost, even though fill is still treated preemptively until the minimum fill level is achieved.

There are five staffing precedence levels among the billets. Top priority jobs should be filled first whenever possible without reducing the overall number of billets filled. In these "priority of fill" objectives, the fill in each of the SPL's is maximized starting from the highest priority billets (SPL 1) to the lowest (SPL 5). The weights on these five objectives are not controllable by the user since they model the natural process of the allocation process itself. Any changes in the policies they incorporate can be captured by altering the SPL's of the billets rather than by changing the fact that higher priority billets are filled first. Priority of fill makes up the second through the sixth objectives.

The seventh objective, "fit", is defined by the E1 Cards which specify acceptable substitutions for each job. In order to be matched to a particular billet, an officer must fulfil at least one of the eligibility requirements defined for that job. The criteria are specified in terms of Primary Military Occupation Specialty (PMOS), additional Military Occupation Specialties (MOS's), grade (rank), experience, sex, duty restriction, and "officer type." Since the best fit has the lowest substitution number, maximizing fit amounts to achieving the minimum sum of fit scores for a given fill. In the prototype, fit defaults to the seventh ordered objective.

The default eighth objective is minimization of Permanent Change of Station costs. If the user desires to emphasize the PCS cost objective more heavily, he may adjust the weight of the cost objective relative to the fit objective by adjusting an α value which is used to make a linear combination of the two weights. This combined "fit-PCS cost" objective is then placed on a single shared arc. The adjustment of the relative weights of the two objectives may be carried as far as a reversal of their priorities so that PCS cost could become the seventh ordered objective, and fit, the eighth. Thus, either may be treated preemptively over the other, or they may be weighted somewhere in between by using a linear combination of their values. Although it is theoretically possible to allow *any* linear combination of the weights of the two objectives, in the model, the user is only given the option of making one or the other preemptive, or of giving them equal weights. Permitting adjustments of their relative weights with more precision would not provide any meaningful information to the decision maker.

The final objective is the sharing of shortages within SPL's. This "proportionality of fill" criterion stipulates that qualified officers should be distributed evenly among the demands with the same skill requirements within each SPL. Although, in the OSGM, this objective is solved right after the priority of fill criteria (objectives two through six), it is modeled last in the prototype. In order to assess the full impact of tradeoffs in changing the fit and PCS cost objectives, it was felt that they should both be solved before shortages were shared among the billets. To restore the precedence of the proportionality of fill objective would require the exchanging of the multiplicative factors used to weight it relative to the linear combination of fit and PCS cost.

B. FORMULATION AND DESCRIPTION OF THE MODEL

1. Mathematical Model

The following mathematical formulation models the capacitated transshipment network that is used to solve the multiobjective allocation problem:

Indices:

$i = 1,\ldots,r$	individual officers (supply nodes)
$j = 1,\ldots,t$	quotas (a specific billet demand node at an MCC)
$k = 1,\ldots,d_j$	index of arcs lying between quota j and quota group h
$h = 1,\ldots,u$	quota groups (node which aggregates all quotas with the same fill priority/SPL)
P	pool node
S	sink node

Set:

Q(h)	set of all quota j's assigned to quota group h, equivalent to the set of all quotas in an SPL

Data:

α	parameter value to determine relative weight of fit to PCS cost in the objective function
a_{iS}	cost of selecting artificial arc from node i to sink S
d_j	demand at the jth quota
e_{hP}	cost of using the arc from quota group h to pool P
f_{ij}	un-scaled fit cost on the arc between supply i and quota j
g_{ij}	PCS cost on the arc between supply i and quota j
v	scaling coefficient for fit used to find c_{ij}
u_{PS}	upper bound for flow from pool node P to sink node S

Cost Functions:

c_{ij}	cost of traveling from supply i to quota j
w^k_{jh}	cost of selecting the kth arc between quota j and the quota group h associated with quota j

Cost Function Equations:

$$c_{ij} = \alpha \times v \times f_{ij} + (1-\alpha) \times g_{ij} \qquad (2.1)$$
$$w^{k}_{jh} = 1/d_j \times (k-0.5) \qquad (2.2)$$

Decision variables:

x_{mn}	flow between any two nodes m and n in the network
x^{k}_{jh}	flow on the kth arc between quota j and quota group h

Formulation:[11]

$$\min \textstyle\sum_i a_{iS} \times x_{iS} + \sum_h e_{hP} \times x_{hP} + \sum_i \sum_j c_{ij} \times x_{ij} + \sum_h \sum_j \sum_k w^{k}_{jh} \times x^{k}_{jh} \quad (2.3)$$

$$\begin{array}{llr}
\text{s.t.} & \sum_i x_{ij} - \sum_k x^{k}_{jh} = 0 \quad \text{for all } j & (2.4) \\
& \sum_j \sum_k x^{k}_{jh} - x_{hP} = 0 \quad \text{for all } h & (2.5) \\
& \sum_h x_{hP} - x_{PS} = 0 & (2.6) \\
& x_{PS} + \sum_i x_{iS} = r & (2.7) \\
& x_{iS} + \sum_j x_{ij} = 1 \quad \text{for all } i & (2.8) \\
& 0 \le x_{hP} \le \sum_{j \in Q(h)} d_j & (2.9) \\
& 0 \le x^{k}_{jh} \le 1 & \text{for all } h,j,k \\
& 0 \le x_{PS} \le u_{PS} & \\
& x_{mn} \text{ integer} &
\end{array}$$

2. Description of the Formulation

The network assignment model used in the prototype is a modification of a formulation presented by Klingman et al. in [Ref. 4] which permits inclusion of multiple pre-emptive criteria in a single network. The formulation used in this thesis is a capacitated transshipment network which includes provision for controlling certain objective function priorities and setting an aspiration level for fill. Figure 3.1 depicts the network model that was used in the prototype.

[11]For notational simplicity, all summations are assumed to be over the entire range of the indexing variable, unless otherwise noted.

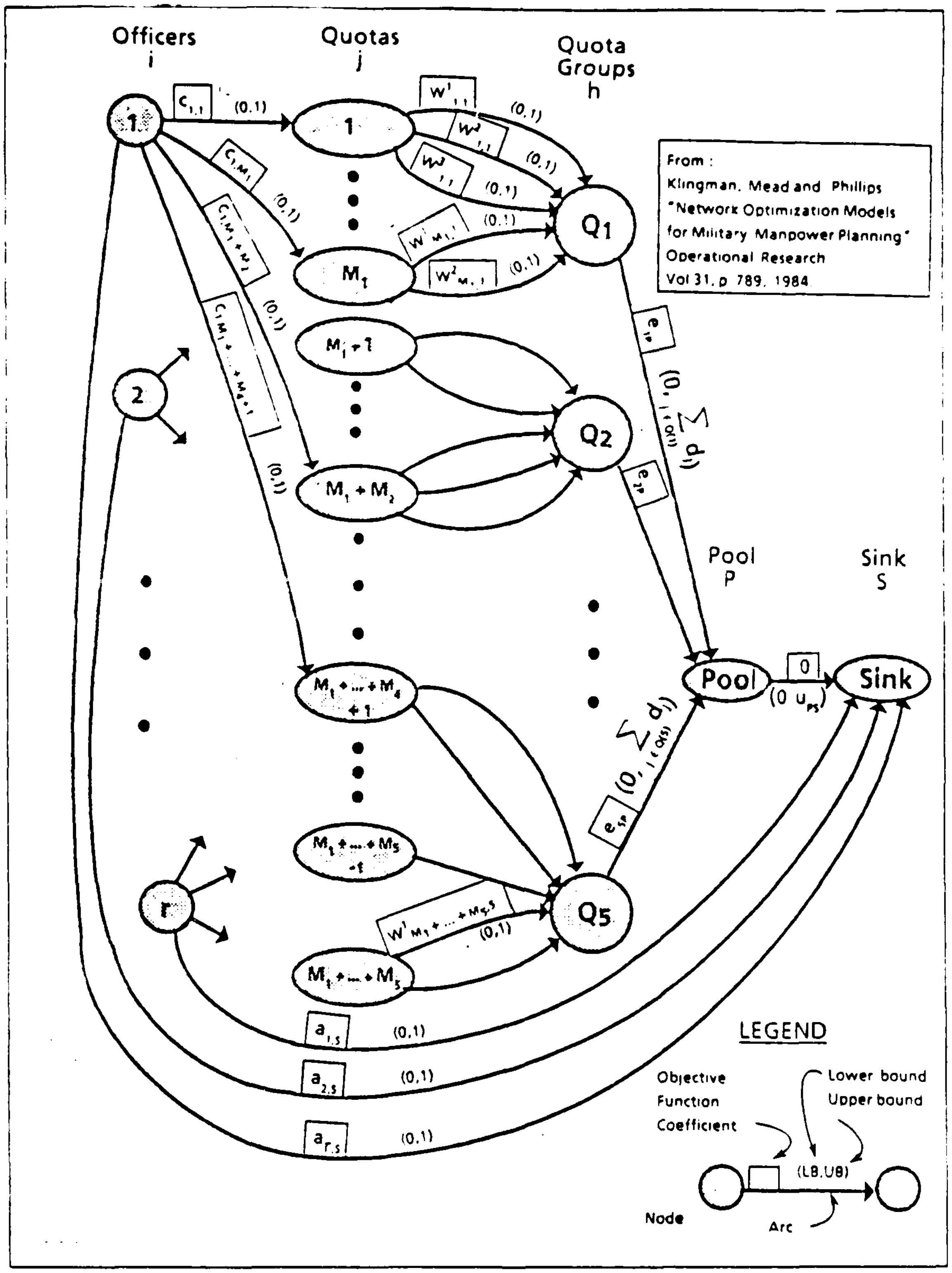


Figure 3.1 Diagram of the Capacitated Transshipment Network.

Structurally, the network is nearly identical to the maximization model presented by Klingman et. al. in [Ref. 4] and [Ref. 5] There are, however, several differences. First, the prototype model provides for the combining of objective function coefficients on some arcs to permit the mixing of the fit and PCS cost objectives. Additionally, an upper bound on fill may be set as an aspiration level. Finally, the formulation presented in [Ref. 4] and [Ref. 5] is a maximization, but because GNET solves the minimum cost problem, the sign of the costs in this formulation has been reversed.

Because individual relocation costs have to be considered in this formulation, it was not possible to aggregate the supply of officers into groups as Klingman et al. suggest. However, in order to reduce the size of the network that must be solved, all billets with the same set of substitution criteria and SPL that are located at the same MCC are grouped into what Klingman et al. call "quotas" [Ref. 4.] Additionally, all quotas with the same SPL are grouped into "quota groups." The nested structure of the resulting network helps to reduce the magnitude of costs that must be used on the arcs.

The nodes on the left of Figure 2-1 represent the supply or "inventory" of those officers eligible to move into jobs and are indexed by i. The arcs coming out of these nodes represent potential matches to the specific jobs which are grouped into quotas (indexed by j) in the second column of nodes. Those quotas having the same priority (SPL) are further grouped into "quota groups" which carry an index of h. Total demand is combined in a "pool" node P which sends the flow to a final "sink" node S which draws the flow through the network.

Flow through the entire network is enforced by constraint equations 2.7 and 2.8. Equation 2.8,

$$x_{iS} + \sum_j x_{ij} = 1 \quad \text{for all } i \qquad (2.8),$$

states that each of the r supplies, indexed by i, must flow either into a quota as an acceptable match, or into the sink. By forcing everyone in the inventory to travel one of these two paths to the demand or sink node, this equation makes possible the maximization of fill by attaching a very high cost (99999) to the arcs going directly to the sink from the inventory as compared to the cost through any of the quotas. Equation 2.7

$$x_{PS} + \sum_i x_{iS} = r \qquad (2.7)$$

ensures that total flow into the sink from the pool and the inventory, equals the supply.

3. Solution of Component Objectives Functions Within the Overall Objective Function

The overall objective function (2.3) is to minimize the weighted sum of the "costs" of sending the inventory (supply) of Marine officers through a capacitated network. The discussion that follows explains in detail the structure of the network, how the component objectives are achieved, and how the problem may be controlled by the user.

a. Maximizing Fill

The primary objective of maximizing the total number of billets filled is met by setting the costs on the "artificial" arcs which run from the supply nodes i to the sink S equal to a very large number. Since fill is to be solved preemptively in practice, the cost for failure to fill a billet was set ten times higher than the difference between the best and the worst fit in any category. This means that in order for a billet to be left unfilled, the gain to the overall solution from leaving that job empty would have to be at least equal to the added value of improving 10 billets from the lowest priority jobs to the highest.

There may be occasions when the decision maker would like to see the effect of sacrificing some amount of fill to improve the fit or the PCS cost in the solution. This can be done without changing the priority of the fill objective by reducing the upper bound, u_{PS}, on the flow from the pool to the sink node. Since the user will know the fill from the previous solution, he may reduce the value of u_{PS} below the previous level in order to give GNET greater flexibility to improve the fit-PCS cost objective. To restore preemptive fill maximization, he need only increase the upper bound to some number greater than the initial solution.

b. Maximizing Fill in High Priority Billets

Once the primary objective of filling the maximum number of billets has been met, the billets are to be allotted so as to maximize the number of billets in the high priority quota groups (SPL's). As with maximizing fill, the idea is to determine costs with a sufficiently large differential to ensure that the priority discipline among the precedence levels is maintained. Because of the relative independence of individual allocations, it was possible to achieve this with relatively low weight differentials between the quota groups and the pool node. The balance constraint for the flow from each quota group to the pool is given in the following equation:

$$\sum_h x_{hP} - x_{PS} = 0. \qquad (2.6)$$

c. Fit-PCS Cost Objective

The objective function coefficients on the arcs from supply i to quota j represent a weighted linear combination of the scaled fit and PCS costs. Both PCS cost and fit are defined such that lower values are better, thus no transformation of their values is required. However, the magnitudes of the measures of fit and PCS cost are quite different. Whereas PCS cost values range from 0 to 8500, fit ranges from 1 to 5. To make their values more comparable, fit was scaled to range from 0 to 8000.

If f_{ij} is the unscaled value of fit, g_{ij} is the PCS cost, h is the scaling factor for fit, and α is a pre-selected weighting coefficient, the objective function coefficient on the arc from supply i to quota j is found by the equation:

$$c_{ij} = \alpha \times v \times f_{ij} + (1-\alpha) \times g_{ij} \qquad (2.1)$$

In the model, α is initially set to 0.99, effectively making fit a preemptive objective over PCS cost. Note that by adjusting α between 0 and 1 before each re-formulation, the decision maker can change the relative priority of the two objectives. To avoid excessive and meaningless manipulation of this parameter in an effort to force the model to attain some preconceived solution by adjustment of α, the decision maker is given only three options. If he chooses to make fit preemptive over PCS cost, then α is set to 0.99. If PCS cost is to be preemptive, then $\alpha = 0.01$. If both objectives are to be weighed equally, $\alpha = 0.50$.

Equation 2.4 shows the flow balance constraint for quotas.

$$\sum_i x_{ij} - \sum_k x^k_{jh} = 0 \quad \text{for all h} \quad (2.4)$$

From this equation and the constraints

$$0 \leq x_{hP} \leq \sum_{j \in Q(h)} d_j \quad \text{for all h} \quad (2.9)$$

it is clear that the total flow into a quota must be limited to the demand at that quota, which also corresponds to the number of arcs out of that quota.

d. Proportional Fill Objective

Having met the first eight objectives, it is desired to make all quotas within quota groups share resources (and hence, shortages) whenever possible. This objective must not reduce either the total number of billets filled, the number filled within each of the quota groups, nor degrade the fit-PCS cost solution. Additional arcs from quotas to quota groups are used to ensure this proportionality of fill within priority levels. Klingman et. al. present a method for also enforcing a disproportionate share policy in [Ref. 5.] However. this option is not developed in the prototype because such a requirement arises infrequently.

Implementation of the objective to proportionately fill within each SPL requires the generation of additional arcs going from the quotas to the quota groups. A single arc for each unit of demand at each quota is generated with an objective function coefficient designed to ensure the sharing of resources. Thus, if a quota has a demand for three billets, there will be three arcs going from the quota to the quota group. This is practical only because most quotas have a demand of between one to five. In the prototype, there were approximately 5,700 arcs going from the 2000 quotas to the four quota groups represented. Since there had to be at least 2,000 arcs in any case, the generation of arcs to meet the proportionality objective resulted in the addition of approximately 3,700 arcs out of a total of 81,000 arcs in the entire network, or about 4.6 percent.

The size of the coefficient that enforces the proportionality constraint depends on the number of demands at the particular quota. Each quota j is associated with a unique quota group h. If d_j is the demand requirement at the j^{th} quota in quota group h, then there will be d_j arcs going from quota j to quota group h. In [Ref. 5] Klingman et. al. develop a coefficient which ensures that the minimum difference in percentage fill among quotas in a given quota group will be achieved. Based on their work, the following coefficient was adopted for the use on the arcs between some quota j and quota group h:

$$w^k_{jh} = 1/d_j \times (k-0.5) \qquad (2.2)$$

The 0.5 constant is used to keep the value less than one.

In the event of severe shortages of certain types of personnel, a straight proportionality distribution scheme may assign several individuals to a quota with a large demand before filling a quota with a single requirement. In practice, however, single billet quotas may represent a critical requirement for a command and should receive exceptional consideration in the distribution of resources. For that reason, the objective function coefficient for the proportionality fill constraint for single demand quotas is rounded down to zero which will cause those billets to be filled first.

Flow balance constraints for quota group h are reflected in constraint equation 2.5.

$$\sum_j \sum_k x^k_{jh} - x_{hP} = 0 \quad \text{for all } h \qquad (2.5)$$

IV. SUMMARY OF RESULTS AND FUTURE ENHANCEMENTS

The first half of this chapter contains a summary of the results obtained from running the prototype model. In order to understand the results, a brief summary of the simplifications included in the model is presented first. The second half of the chapter consists of a two part discussion of future enhancements. The first section presents changes to the prototype which should be made as part of turning it into a full scale model. into a full scale model. Next are listed some possible applications of the system in manpower policy or budget analysis.

A. SUMMARY OF RESULTS

In order to interpret the implications of the results, it will be helpful to review the simplifications made in the model. The following simplifications were made in developing the system:

1. The prototype only considered officers with PMOSs which are related to aviation in order to reduce the size of the problem. This is about 35-40 percent of the Marine Corps.
2. Only substitutions containing specific MOSs and grades were considered. All substitutions indexed by officer type, OCC field, or AMOS were omitted in order to reduce the size of the problem. This effectively reduced the number of billets that were considered, since many staff billets do not require a specific MOS and are defined by officer type, OCC field, or AMOS. (See discussion in Section B.1.b.(1).)
3. No adjustment is made to the inventory to account for accessions in order to simplify the model.
4. No cost tables were developed to accurately estimate costs. Rather, a constant (2000) was added to the distance between the cost centers of the two MCCs involved in any match, and this number was used as the PCS cost. Development of cost tables is beyond the scope of this thesis.

Because of the above simplifications put into the model, it does not consider every possible person-to-billet match. Therefore, the only legitimate analysis of the solution lies in checking the reasonableness of the output and whether the changes to the various parameters caused sensible adjustments in the subsequent solution.

Some detailed analysis was performed on two small networks whose optimal solution was known. Because these two networks were input as sorted lists of potential matches (equivalent to the output from the sorting routine), no manipulation of the

TCF was performed. However, the correct formulation was generated by NET-GENX and the optimal solution was achieved in all cases, including when the fill aspiration level and Fit-PCS cost weights were changed. When the PCS cost and fit data in the two test networks was changed, a correct formulation was produced and the correct solution was once again attained.

The prototype was also run using the inventory and billets relating to aviation and aviation support MOSs. The inventory of 7,116 Marines was matched to a comparably sized demand. Because only those E1 Cards containing a specific MOS and grade were considered, the number of billets available to be filled was reduced by about 22 percent to 5,600. A total of 73,175 people-to-billet matches were generated. This grew to 78,784 arcs in the capacitated transshipment network with 73,175 arcs from individuals to quotas, 5,600 from quotas to quota groups, four from quota groups to the pool, and one from the pool to the sink. The solution from the first pass through the system is summarized in Table 3.

TABLE 3
SUMMARY OF INITIAL SOLUTION OF THE PROTOTYPE

SPL	Demand	Fill	Pct Fill	PCS Cost	Ave Fit
0	0	0	0	NA	NA
2	136	112	0.824	NA	NA
3	4381	3197	0.730	NA	NA
4	15	8	0.533	NA	NA
5	1068	481	0.450	NA	NA
TOTAL	5600	3798	0.678	3344	2.67

Note that there were no demands for any SPL 1 billet. There are no SPL 1 billets among the aviators, so none appeared in the solution. Note also that most of the billets are SPL 3 or 5, which reflects the actual current ordering of Marine Corps priorities in peace time. Out of 5600 billets defined in the problem, 3798 were "filled" by having individuals in the inventory matched to them, for a fill percent of .678. The fill percentages in the SPL 2 to 5 billets seem to indicate that the model was in fact filling the higher SPL billets first. As was already mentioned, the average cost figure was derived from an artificially constructed cost table, and cannot be used for cost analysis yet. Also, PCS cost and average fit were not extracted from the solution in order to reduce the complexity of the prototype.

The low fill percentages, especially in the last two SPLs are reasonable. All those expected to retire and all reservists whose contractual obligation ended during the period considered in the model are deleted, and no adjustment is made to increase the inventory to account for accessions. Also, because the system does not consider any matches involving E1 Cards with officer types or non-specific PMOSs, all those fixed officers assigned to billets described by such E1 Cards who do not get allocated are not counted as filling any billet. About 25 percent of all billets fall into these two categories, most of which are SPL 4 and 5 jobs. When these factors are considered, the percentage of fill figures appear much more reasonable.

Additionally, the fill percentages among the SPLs suggest that the SPL fill objectives were solved preemptively. To test this, the weights on each SPL objective were individually increased by a factor of ten to see if the ordering of fill percentages would remain the same. In all cases, the solution to the problem did not change, indicating that the weights placed on the arcs between the quota groups and the pool node were adequate to insure that the SPL objective functions are solved in the desired order.

Several changes were made to the parameters of the problem to test the operation of the programs that control user modification of the problem.

The first adjustment that was made consisted of reducing the fill to improve the fit-PCS cost objective. Since the initial solution yielded a total fill of 3,798, the aspiration level for fill was lowered to 3,600. The solution to the modified problem showed an improvement in both PCS cost and Fit. The average PCS cost dropped from $3344.00 to $2821.00, and the average fit changed from 2.67 to 2.41. It is possible that both may not improve in all cases. In another run, using a slightly different inventory (a subset of the one used in the prototype), a reduction in fill resulted in an improvement in the fit (the high priority objective) and a degradation of the PCS cost solution. This is reasonable if the new reduction in fill opened up more possibilities of improving fit at the expense of the PCS cost objective.

Next, the order of the PCS cost and fit priorities was reversed, and the aspiration level was removed from fill. In this case, however, there was no change in the solution, indicating that there were no ties remaining for the solver to improve the last two objectives. In the two smaller problems with known optimal solutions, the coefficients were set to ensure that there would be ties, and the solution indicated an improvement in PCS cost, as expected.

Finally, changing eligibility for transfer through the TCF adjustment factor resulted in the expected changes in the fixed and free inventories. Increasing the TCF increased the number of fixed personnel, while decreasing it increased the number of free personnel. This is exactly what would be expected since increasing the TCF causes people to stay in their billets longer, thus reducing the turnover rate. Reducing it should increase the number of movers since people will move more often if tour lengths are decreased.

It is not possible to precisely estimate the impact of a change in one of the factors on the overall solution because of the many interactions involved, which is why the system is needed in the first place. However, it is possible to determine if the resulting change is reasonable. In all cases that were attempted on the prototype, the effect on the solution of a change in one of the user-controlled parameters was reasonable.

B. FUTURE ENHANCEMENTS

The prototype system presented in this thesis does not exhaustively include all specific factors affecting the allocation process. Instead, it is designed to permit the incorporation of as many broad categories of factors (such as extraction rules, matching criteria, and objective priorities) as possible through the use of flexible data extraction and network formulation. In this section, a number of possible future enhancements are listed which could improve the speed, realism, or completeness of the system. Also, some examples of possible applications of the capabilities of the resulting model are discussed.

1. Possible Improvements in Speed, Realism and Completeness

a. Improvement in Speed of Execution

The primary goal of this thesis is to demonstrate the feasibility of the basic design concept, not to minimize execution time. Numerous extra files were added, both to assist in following the internal processes of the program and in anticipation of files that might be helpful to possible future users.[12] Therefore, solution times should not be used as an absolute gauge of the performance of the basic model. Nonetheless, there is an obvious legitimate interest in reducing execution time. In this section, the

[12]For example, in FREE-FIX one of the files that is generated contains all those who are expected to retire or leave during the next year because of the end of their reserve contract obligation.

solution times achieved in the prototype will be used as rough benchmarks for the areas that promise the greatest return for improvements in efficiency or speed. Table 4 lists the programs and the execution times achieved when the model was run using all Marine Corps aviation officers and billets (Approximately 7,100 supplies and 5,600 demands).

TABLE 4

SUMMARY OF COMPONENT PROGRAM SOLUTION TIMES

Program Name	Language	Run Time (CPU sec)	Percent of Total Time	Rank (=1 is slowest)
INITLIZE	FORTRAN	0.07	0.01	19
FREE-FIX	SAS	47.67	6.99	4
ADJ-LIST	FORTRAN	1.72	0.02	18
ENTRY-PT	FORTRAN	0.33	0.04	17
ROLINV	SAS	3.99	0.50	11
XPASR	SAS	1.56	0.19	15
C1ASR	SAS	3.59	0.46	12
INVNTRY1	SAS	7.23	0.91	10
ADJ-LIST	SAS	2.63	0.33	14
EXCESS	FORTRAN	3.52	0.44	13
ASRE2A	SAS	13.06	1.64	9
ASRE2B	SAS	15.00	1.88	7
E2ASRE1A	SAS	14.11	1.77	8
E2ASRE1B	SAS	16.52	2.08	6
MATCH-AL	FORTRAN	39.80	5.00	5
BIGSORT	SAS	105.37	13.24	2
NET-GENX	FORTRAN	59.31	7.45	3
GNETBX	FORTRAN	459.00	57.66	1
SUMMARY	FORTRAN	1.50	0.19	16
TOTAL	NA	796.08	100.00	NA

There are two ways that offer good promise in speeding up the execution time of a single pass through the system: solution of each objective function successively, and conversion of all programs to Fortran.

(1) *Iterative Solution of the Network Formulation.* Almost sixty percent of the run time is taken up in the solution of the network formulation in GNETBX. Recall that all priorities are solved in a single pass through the use of weights. The research of Klingman et al. [Ref. 5] suggests that solution of each of the objectives in order of its priority might be computationally faster. After each component objective is optimized, the optimal flow value would be placed on the appropriate arc as a lower

bound on the flow. This approach can only be used when the optimal value of the objective function can be expressed as the flow along a single arc. If an objective function requires more than one arc, then setting the lower bounds on those arcs to the optimal flows will overly constrain all subsequent objective functions. Thus, this technique can only be applied to those priority objectives up to the first objective function requiring more than one arc. All subsequent objectives must be solved using some other method, such as weights.

This approach could be built into the system by modifying the way that NET-GENX determines arc costs and bounds, and changing the EXEC program so that the component objective function solution from GNETBX would continue to be sent back into NET-GENX until all objectives had been solved, there were no more ties, or the program reaches the first objective that is no longer expressible as the flow on a single arc.

(2) *Conversion of SAS Programs to Fortran.* SAS was used extensively in the programming of the system because it is a convenient language to use for manipulation of data, and because of its simple sorting, merging, and matching routines. Unfortunately, unlike Fortran, SAS lacks the capability to read in multiple data sets simultaneously. This could speed up the ASRE2A, ASRE2B, E2ASRE1A, and E2ASRE1B programs which must read some large data sets several times because SAS cannot read part of a second data set while another one has not yet been completely read. Also, interactive use of SAS involves a large amount of overhead even for small data sets. Finally, it might be possible to improve the 105 seconds of CPU time required to sort the 78,784 people-to-billet matches in BIGSORT by use of a specially tailored Fortran sorting routine. By conversion of the entire system to a single language, speed and clarity might be increased. All file definitions and compilation could be done at once, and there would be no need for an EXEC control program, since the entire system could be controlled within a Fortran main program.

Another change that would reduce the run time in an actual implementation is the elimination of any of the extra files that were generated in the prototype, once it is determined which of those files will not be useful to the decision maker. Reduction of the run time of the system would permit the inclusion of more options to make the model more accurate, versatile, and "user friendly."

b. Improvement in Realism and Completeness

Since the prototype contains the functional structure rather than the complete implementation of the concept, many assumptions and simplifications were made to arrive at a simpler system. These should be checked one by one and, if necessary, removed in any full scale design of the model. Additionally, the prototype model includes a number of programming simplifications which should be eliminated in the full scale version. These include:

1. Limited error checking on the input data
2. Limited error checking on the user inputs to the interactive part of the model to prevent program termination due to a typing error by the user.
3. Failure to calculate the average PCS cost and fit within each SPL.

(1) *Develop Pre-Processor to Check Input Files.* Most of the programs contain checks to filter out bad data on the input files. They could be streamlined somewhat if certain checks on the integrity and completeness of the input files were run as they are being read in from Marine Corps files the first time. This was not done in the prototype, because most of the potential problems with the data were determined while the programs were being written, and it was more convenient to perform the check at the point where the flaws were detected.

(2) *Develop Enhanced Solution Presentation and Options.* The summary of the solution which is presented to the decision maker is collected from several files which contain pertinent data on the flows through the network as well as the parameters used to define the formulation. In the prototype, the statistical summary presented to the user is very rudimentary. In a full scale model, the user will want the option of exploring the distributions of resources in the allocation solution in much greater detail than given in the prototype, including interactive selection of different statistical summaries of the solution. For example, the user may desire to see a breakdown of the solution by OCC fields, MOSs, MCC's, or broad geographic areas. By extracting additional information from GNETBX and the files containing the user-defined parameters, the solution presentation could be greatly enhanced. Possible areas of interest to a decision maker include the staffing within certain OCC fields, MOSs, grades, and MCCs. Finally, addition of written reports to summarize the various solutions would be desireable.

(3) *Generate All Possible Matches.* There are three primary ways of matching individuals to billets based on the type of E1 Card used for a given billet.

One method is indexed on the PMOS and grade, another on AMOS or OCC field and grade, and the third on officer type and grade. The prototype generates matches only for the first type of E1 Cards, those which contain an explicit PMOS along with six other criteria: grade, AMOS1, AMOS2, experience, sex, and duty restriction. By matching with those E1 Cards, the model demonstrates the basic logic and efficiency of the matching program while reducing the size of the matching routine in the prototype, since it would take more than 1000 additional lines of Fortran code to implement the last two E1 types. However, this ignores about twenty-five percent of the E1 Cards, which is clearly unacceptable in an implementation where all matches must be found. Therefore, the remaining E1 cards must be matched in any full scale operational system. The program MATCH-AL is written so that appropriate subroutines could be inserted to accomplish this without disturbing the flow of the program.

(4) *Account for Frictional Losses.* In the prototype, the inventory is reduced by the number of officers who are expected to retire or whose reserve contract expires during the period of the model run. The number of officers who are actually considered for allocation should be further reduced to account for attritions, and non-availability due to training, transfer, or other reasons. Estimates of these losses, called Prisoners, Patients, Transients, and Trainees (P2T2) losses, are made by Headquarters Marine Corps and are contained in a Dictionary file called WKE3 CRD. The "E3 Cards" which make up this file list the reduction in numbers of Marines with a particular MOS and grade due to P2T2 losses. No such adjustment to the inventory was made in the prototype., P2T2 losses could be accounted for in two ways. First, the MOS grade combinations listed in the E3 Cards could be treated as demands and added to the ASR with an SPL of 0. This would cause them to be filled first, although there is a chance that some of the imaginary P2T2 "billets" would not be "filled" just as some billets in every SPL may go unfilled. The other way is to reduce the indicated MOS grade population using a random sampling procedure.

(5) *Account for Accessions.* The "F3 Cards" contained in the file WKF3 CRD give the numbers of accessions to the inventory by grade and MOS. These could be included by addition of the number and type of individuals indicated on the F3 Card to the inventory at some appropriate MCC.

(6) *PCS Cost Data.* The program MATCH-AL attaches a PCS cost to each person-to-job match using a matrix indexed on the Cost Centers of the individual and the billet. The PCS cost values contained in the matrix are equal to the distance

between the Cost Centers plus an arbitrary constant (2500) which was chosen to make the values seem reasonable. Actual cost tables need to be developed before PCS cost can be included in an actual implementation. If a PCS cost matrix that is indexed on grade as well as Cost Center is developed, the model could be modified to capture even more accurate relocation costs.

(7) *Redefine Reassignments.* In the prototype, reassignments are permitted only to other billets at the Marine's current MCC. In practice, however, they could be made to any other MCC within a 50 mile radius of the current MCC. Additionally, there are certain regions, defined by Marine Corps directive, within which reassignment is permitted even if the distances between MCCs exceeds 50 miles. This was not done in the prototype because there is no readily available file which groups the MCCs into such regions. The Cost Code Centers which are used to estimate costs are not suitable surrogates for these regions since they are not sufficiently restricted in range to ensure the 50 mile restriction. (In some cases, MCCs within the same Cost Center are located more than 500 miles apart.) Development of an array which determines if reassignments between pairs of MCCs is legitimate was beyond the scope of this thesis, but would be a valueable enhancement to the system.

2. Applications

There are certain sections within some of the functional areas which could be modified to permit a variety of future applications that would enhance the model's usefulness as an analysis tool.

a. Data Extraction

In the prototype, the TCF adjustment is applied equally across all billets. Allowing for different TCF adjustment factors within OCC field groups or MCC regions (eg. all overseas commands) would enable the decision maker to analyze specific policy proposals with regard to eligibility and tour length requirements.

The system presently determines fixed and free inventories by applying the TCF eligibility criteria to all individuals who are eligible to move one year from the date that the model is run. This is called the "window" of eligibility. By allowing an adjustable window so that the user may look at some time frame other than one year, the decision maker can evaluate the staffing goal using different planning horizons.

In the list of simplifications, in Section A it was mentioned that no adjustment is made to account for accessions. However, such a capability does exist in the OSGM through the use of files that estimate the numbers of particular MOS-grade combinations that are expected to enter the Marine Corps during the window of the eligibility. These estimates are used to increase the inventory in the appropriate MOS and grade categories before the allocation model is run. These files could be adapted to permit the addition of any number of certain MOS and grade combinations. By allowing the user to generate additional people of any desired description (ie. MOS, grade, etc.) the system would provide the capability to analyze the staffing fill if the Marine Corps were to increase the training quotas for certain groups or try to increase retention within certain MOSs using some incentive such as bonuses.

b. Data Preparation

In this thesis the adjustment of supply and demand is based solely on the existing inputs from Marine Corps files, and is updated by a file that contains recent changes to the ASR. Addition of a module to alter the file used in the update could create a capability for the decision maker to evaluate the impact of hypothetical changes to the structure as well. Certain common unit organizational structures and individual billets could be pre-input into the WKC1 CRD file and turned "on" or "off" using binary flags which could be set by the user. Normally these flags would be off, meaning no changes to the structure. But if the user wished to test the staffing goal with, for example, an additional three battalions in the Second Marine Division, he could turn on the appropriate flag and designate where the addition is to occur. When combined with the ability to change the numbers of various officer types, this tremendously expands the potential analysis capability of the model.

c. Problem Formulation

In the prototype, only PCS cost and fit may have their objective priorities reversed, and only fill may be reduced by setting an aspiration level. Theoretically, all objectives could have the capability of changing priorities, and those that can be captured by the flow on a single arc may have aspiration levels. In Chapter III, the concept of using pairwise weighted costs to give the user control over the relative priorities of pairs of objective functions was presented. The same principle could be extended to include other pairwise matchings of related factors. By using binary "flags" in the control program to turn the costs attached to these pairs of criteria "on" or "off", the effect of tradeoffs among numerous combinations of objectives could be modeled. By grouping related pairs of criteria and placing the weighted coefficients from their linear combination on an extra set of arcs connecting the appropriate families of nodes, a large number of additional related policy or criteria tradeoffs could be modeled and tested. For example, there are many factors related to the matching of people to billets which could be quantified in the matching program, such as grade or MOS substitutions, experience flags, and time since last overseas deployment. These measures would be scaled to be roughly comparable in size and fed to the network generation program. Once the appropriate on/off flags are set by the user, the problem could be formulated to evaluate the impact on the staffing goal of such policy alternatives as loosening grade/MOS substitution criteria or changing the present policy on time between overseas assignments, rather than just comparing the tradeoffs between fit and PCS costs, as is done in the prototype.

V. CONCLUSIONS

The system presented in this thesis demonstrates the feasibility of building an integrated decision support system that uses readily available data files and existing software to analyze the impact of policy and budgetary changes on staffing levels in the Marine Corps. The system design incorporates a number of practical features which make it an excellent framework on which to build a larger scale implementation.

The results of running the prototype using only aviation officers seemed to indicate that the multiple objectives incorporated into the model were being met in the desired order.

The system utilizes data files that already exist, and would therefore not require the development, validation, or maintenance of a new set of files. The modular structure of the system facilitates the addition of enhanced capabilities without making any fundamental structural changes to the system.

All of the system programs except for the VS EXEC controller routine are written in languages which are available on the Marine Corps mainframe at Quantico, Virginia. The SAS programs can be directly imported to the Quantico computer, and the Fortran programs require only those changes necessary to convert from VS Fortran to Fortran 77. The functions performed by the control program can be programmed into a TSO EXEC, which is very similar is syntax and structure to a VS EXEC.

The system may be used to answer a wide variety of questions by adjusting policy related parameters. This includes changing the rules determining eligibility for transfer, changing the priority of certain objectives, and adjusting the aspiration level for total fill. Thus, the user may ask "what-if" questions regarding numerous policy and budgetary proposals.

In its role as a prototype, the system was not designed to provide an exhaustive exploration of possible outcomes even within the PCS cost area. Rather, the model demonstrates the feasibility of using a control module, various data extraction and pre-processing modules, and a set of programs which permit variable formulation of a network assignment problem, to provide a decision maker with the ability to evaluate the staffing levels in varying scenarios. Further development of the various concepts applied in this prototype are required before any full scale implementation can be constructed. However, the basic design of the system offers the potential to improve the ability of Marine Corps decision makers to wisely manage their limited manpower and budgetary resources in the years ahead.

APPENDIX A
THESIS EXEC

1. CMS EXEC PROGRAM TO PROVIDE OVERALL CONTROL OF THE SYSTEM

```
***********************************************************************
*                                                                     *
*         *  *  *  *   PROGRAM NAME:  THESIS EXEC   *  *  *  *        *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*                  * * *   OVERVIEW AND PURPOSE    * * *              *
*                                                                     *
*      THIS IS A CMS EXEC PROGRAM DESIGNED TO PROVIDE OVERALL CONTROL *
* OF THE PROGRAM FLOW THROUGH THE DECISION SUPPORT SYSTEM.  IT DIRECTS *
* THE COMPILATION OF FORTRAN PROGRAMS AND DECLARES FILEDEFS AT THE    *
* APPROPRIATE TIME.  IT  DIRECTS THE  PROGRAMS TO EXECUTE IN THE PROPER *
* SEQUENCE.  AT THE END OF EACH SOLUTION SUMMARY, IT GIVES THE USER THE *
* OPPORTUNITY TO RUN THROUGH THE SOLUTION PROCESS AGAIN, AND DIRECTS  *
* APPROPRIATE PROGRAM TO EXECUTE DEPENDING ON WHICH PARAMETERS THE USER *
* HAS CHANGED.                                                        *
*                                                                     *
***********************************************************************
-ALP
&TIME ON
&TIME RESET
&FN  = INITLIZE
-H FORTVS &FN
-RUN
FILEDEF 1 DISK ALPHA DATA A
FILEDEF 2 DISK UPPR-BND DATA A
FILEDEF 3 DISK BESTNUM DATA A
FILEDEF 4 DISK TCF-ADJ DATA A
&TIME RESET
LOAD &FN (START
&TIME RESET
EXEC SAS FREE-FIX
&TIME RESET
&COMMAND ERASE TEMP INVENTRY A
&TIME RESET
EXEC SAS SAVEFIXD
&TIME RESET
&FN  = ADJ-LIST
FORTVS &FN
FILEDEF  8 DISK USMC MOVRSUP A
FILEDEF  9 DISK ADJ-MOVR ARRAY A  (PERM RECFM F LRECL 80
FILEDEF 10 DISK USMC NONMSUP A
FILEDEF 11 DISK ADJ-NMOV ARRAY A  (PERM RECFM F LRECL 80
&TIME RESET
LOAD &FN (START
&TIME RESET
&FN  = ENTRY-PT
FORTVS &FN
FILEDEF  9 DISK ADJ-MOVR ARRAY A  (PERM RECFM F LRECL 80
FILEDEF 12 DISK MOVR-EP ARRAY A  (PERM RECFM F LRECL 80
FILEDEF 11 DISK ADJ-NMOV ARRAY A  (PERM RECFM F LRECL 80
FILEDEF 13 DISK NMOVR-EP ARRAY A  (PERM RECFM F LRECL 80
&TIME RESET
LOAD &FN (START
&TIME RESET
EXEC SAS ROLINV
&TIME RESET
EXEC SAS XPASR
```

```
&TIME RESET
EXEC SAS C1ASR
&TIME RESET
EXEC SAS INVNTRY1
&TIME RESET
EXEC SAS ADJ-LIST
&TIME RESET
&FN  = EXCESS
FORTVS &FN
FILEDEF 14 DISK ADJ-LIST EXCESS A
FILEDEF 15 DISK ADJ-LIST ALL-BIL A
FILEDEF 16 DISK NO-ASR CARD A
FILEDEF 17 DISK ASR-INV TOT-MRGE A
FILEDEF 18 DISK FREE ASR A
FILEDEF 19 DISK NO-XCESS SUP-DEM A
FILEDEF 20 DISK EXCESS PERSONL A
&TIME RESET
LOAD &FN (START
&TIME RESET
EXEC SAS ASRE2A
&TIME RESET
EXEC SAS E2ASRE1A
&TIME RESET
EXEC SAS ASRE2B
&TIME RESET
EXEC SAS E2ASRE1B
&TIME RESET
&FN  = MATCH-AL
FORTVS &FN
FILEDEF 21 DISK MOVR-DEM INPUT A1 (LRECL 80
FILEDEF 22 DISK DEBUG OUTPUT A1
FILEDEF 23 DISK EASY-MOS MATCH A1
FILEDEF 24 DISK HARD-MOS MATCH A1
FILEDEF  8 DISK USMC MOVRSUP A1
FILEDEF 12 DISK MOVR-EP ARRAY A1
FILEDEF 26 DISK MATCH OUTPUT A1 (PERM RECFM F LRECL 35
FILEDEF 10 DISK USMC NONMSUP A1
FILEDEF 27 DISK NMOV-DEM INPUT A1
FILEDEF 25 DISK NON-MOS MATCH A1
FILEDEF 28 DISK COST-CTR DIST-MAT A1
FILEDEF 29 DISK TEST-OF COST-OUT A1
FILEDEF 30 DISK SUP-SIZE DATA A1
FILEDEF 01 DISK ALPHA DATA A1
FILEDEF 13 DISK NMOVR-EP ARRAY A1
&TIME ON
&TIME RESET
LOAD &FN (START
&TIME RESET
EXEC SAS BIGSORT
&TIME RESET
-WTS
&FN  = NET-GENX
FORTVS &FN
FILEDEF 30 DISK SUP-SIZE DATA A1
FILEDEF 01 DISK ALPHA DATA A1
FILEDEF 31 DISK SORTED RAW-ARCS A1
FILEDEF 02 DISK UPPR-BND DATA A1
FILEDEF 32 DISK DEBUG ARC-FILE A1 (PERM RECFM F LRECL 100
FILEDEF 33 DISK HOPE-FUL OUTP A1 (PERM RECFM F LRECL 130
FILEDEF 34 DISK GNET INPUT A1
FILEDEF 35 DISK NET-INFO DATA A1
FILEDEF 36 DISK TNUMQG DATA A1
&TIME RESET
LOAD &FN (START
&TIME RESET
&FN  = GNETBX
FORTVS &FN
FILEDEF 34 DISK GNET INPUT A
FILEDEF 37 DISK GNET OUTPUT A (RECFM F LRECL 120 BLOCK 120
FILEDEF 38 DISK SUMMARY INFO1 A
FILEDEF 36 DISK TNUMQG DATA A
FILEDEF 03 DISK BESTNUM DATA A
FILEDEF 39 DISK BSTNUMB DATA A
LOAD &FN (START
&TIME RESET
&FN  = SUMMARY
FORTVS &FN
FILEDEF 35 DISK NET-INFO DATA A
FILEDEF 38 DISK SUMMARY INFO1 A1
FILEDEF 30 DISK SUP-SIZE DATA A1
FILEDEF 40 DISK TEST-SUM OUT A
FILEDEF 02 DISK UPPR-BND DATA A1
FILEDEF 03 DISK BESTNUM DATA A1
FILEDEF 41 DISK UPPERBND DATA A1
FILEDEF 01 DISK ALPHA DATA A1
FILEDEF 42 DISK ALPHAX DATA A1
FILEDEF 04 DISK TCF-ADJ DATA A1
FILEDEF 43 DISK TCFXADJ DATA A1
FILEDEF 44 DISK OPTION DATA A1
&TIME RESET
LOAD &FN (START
&TIME RESET
&COMMAND ERASE UPPR-BND DATA A
&COMMAND ERASE ALPHA DATA A1
&TIME RESET
EXEC SAS CHG-DATA
&TIME RESET
&COMMAND ERASE UPPERBND DATA A
&COMMAND ERASE ALPHAX DATA A1
&TIME RESET
-RAK
&TYPE DO YOU WISH TO RE-RUN THE PROBLEM WITH NEW WEIGHTS / BOUNDS (W), OR
&TYPE THE ENTIRE PROBLEM INCLUDING POLICY CHANGES?  (P)
&TYPE OR,DO YOU WISH TO QUIT? (Q)
&READ VARS &RCOMP
&IF &RCOMP EQ W &GOTO -WTS
&IF &RCOMP EQ P &GOTO -ALP
&IF &RCOMP NE Q &GOTO -RAK
&EXIT
```

APPENDIX B
INITLIZE FORTRAN

1. PROGRAM TO INITIALIZE PARAMETER VALUES

```
************************************************************************
*                                                                      *
*                    PROGRAM NAME:  INITLIZE FORTRAN                   *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                         OVERVIEW AND PURPOSE                         *
*                                                                      *
*   THIS PROGRAM INITIALIZES THE VALUES OF CERTAIN VARIABLES LOCATED    *
* IN INPUT FILES WHICH WILL BE READ IN DURING THE FIRST PASS THROUGH   *
* THE SYSTEM.  THEY MIGHT LATER BE MODIFIED BY THE USER IN SUBSEQUENT  *
* PASSES THROUGH THE MODULES IN THE SYSTEM.                            *
*                                                                      *
*      IN ORDER TO ALLOW THE USER TO CHANGE THE WEIGHTS ON THE FILL    *
* FIT OBJECTIVES, THE ASPIRATION LEVEL FOR TOTAL FILL, AND THE TOUR    *
* CONTROL FACTOR, IT IS NECESSARY TO ESTABLISH FILES TO CONTAIN THE    *
* USER-DEFINED PARAMETERS.  HOWEVER, THESE PARAMETERS ARE NOT INPUT    *
* BY THE USER UNTIL AFTER THE FIRST PASS THROUGH THE NETWORK SOLVER.   *
* THEREFORE, IN ORDER TO MAKE THE PROGRAMS WHICH READ THE USER-INPUT   *
* PARAMETERS BE ABLE TO SOLVE THE PROBLEM BEFORE THE USER MAKES ANY    *
* DESIRED CHANGES, IT IS NECESSARY TO INITIALIZE THE FILES TO DEFAULT  *
* VALUES WHICH WILL CAPTURE CURRENT POLICIES.  THUS, THE TCF ADJUST-   *
* MENT CONSTANT IS SET TO 0, MEANING PRESENT TCF POLICIES HOLD IN THE  *
* FIRST RUN OF THE SOLVER.  THE VALUE OF ALPHA WHICH DETERMINES THE    *
* RELATIVE PRIORITIES OF THE FIT AND PCS COST OBJECTIVES IS SET TO     *
* 0.99, MEANING THAT THE FIT OBJECTIVE IS MORE HEAVILY  WEIGHTED THAN  *
* THE PCS COST OBJECTIVE.  FINALLY, IT IS DESIRED TO LET THE SOLVER    *
* MAXIMIZE TOTAL FILL WITHOUT LIMITING IT TO SOME ASPIRATION LEVEL.    *
* THUS, THE FILL TARGET IS SET TO 30,000 WHICH IS MUCH HIGHER THAN     *
* CAN POSSIBLY BE ATTAINED SINCE THERE ARE ALWAYS LESS THAN 25,000     *
* PEOPLE IN THE INVENTORY.                                             *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                           FILE DEFINITIONS                           *
*                                                                      *
*    FILEDEF    FILE IDENTIFICATION            PURPOSE                 *
*       1         ALPHA DATA A1        FIT/PCS COST PARAMETER FILE     *
*       2         UPPR-BND DATA A1     TOTAL FILL PARAMETER FILE       *
*       3         BESTNUM DATA A1      RECORDS MAX FIX ACHIEVED        *
*       4         ADJST TCF-DATA A1    TCF ADJUSTMENT PARAMETER FILE   *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                          DEFINITION OF TERMS                         *
*                                                                      *
*   ALPHA     USED TO ADJUST RELATIVE WEIGHT BETWEEN THE FIT AND       *
*             PCS COST OBJECTIVES                                      *
*   BOUND     UPPER BOUND ON FLOW FROM THE POOL TO THE SINK NODES      *
*             IN THE CAPACITATED TRANSSHIPMENT PROBLEM                 *
*   LSTBST    THE BEST (LARGEST NUMBER) FILL OF ALL PREVIOUS SOLUTIONS *
*   MARKR     USED TO INDICATE HOW MANY TIMES THE USER HAS GONE        *
*             THROUGH THE SOLUTION LOOP.                               *
*   TCFADJ    TCF ADJUSTMENT CONSTANT                                  *
*                                                                      *
************************************************************************
      INTEGER MARKP BOUND,IMARKR
      REAL ALPHA
* INITIALIZE THE ITERATION MARKER AND THE ALPHA VALUE USED TO WEIGHT
* THE FIT AND PCS COST OBJECTIVE FUNCTIONS.
```

```
      MARKR = 0
      IMARKR = 0
      ALPHA = .99
      WRITE(91,101) IMARKR,ALPHA
101   FORMAT(I1,1X,F4.2)
* INITIALIZE THE UPPER BOUND ON THE FILL ARC IN THE NETWORK
      BOUND = 30000
      WRITE(92,102) BOUND
102   FORMAT(I5)
* INITIALIZE NUMBER USED TO COMPARE THE FILL IN SUCCESSIVE SOLUTIONS
      LSTBST = 3798
      WRITE(93,103) LSTBST
103   FORMAT(I5)
* INITIALIZE TOUR CONTROL FACTOR ADJUSTMENT TO ZERO
      TCFADJ = 0
      WRITE(94,104) TCFADJ
104   FORMAT(I2)
      STOP
      END
```

APPENDIX C
FREE-FIX SAS

1. PROGRAM TO EXTRACT AND BEGIN PROCESSING OF INVENTORY

```
**************************************************************************
*                                                                        *
*           * * * *   PROGRAM NAME:  FREE-FIX SAS   * * * *              *
*                                                                        *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
*                                                                        *
*                   * * *  OVERVIEW AND PURPOSE  * * *                   *
*                                                                        *
*    This program reads in the entire inventory (supply) of officers     *
* and performs most of the initial processing on the data. Its           *
* functions include the following tasks:                                 *
*    1. Read in the inventory from USMC personnel files                  *
*    2. Recode certain variables into 0, 1, or 2 integer values.         *
*    3. Assign officer type to each officer using the B1 cards from      *
*       the Dictionary.                                                  *
*    4. Read in Tour Control Factor (TCF) adjustment from TCF-ADJ        *
*       DATA file and calculate new TCFs.                                *
*    5. Add cost center codes to each individual to indicate his         *
*       present location.                                                *
*    6. Generate file of non-movers. (USMC NMOVSUP)                      *
*    7. Generate file of movers. (USMC NONMSUP)                          *
*    8. Remove retirees and those reservists who will be getting out.    *
*    9. Sort and output the mover and non-mover files.                   *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                  * * *  FILE DEFINITIONS  * * *                        *
*                                                                        *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
CMS FILEDEF DATAIN1 DISK WORKING INVENTRY A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN2 DISK TEMP INVENTRY A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN3 DISK WKB1 CRD A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN4 DISK CCC-MCC CONVERT A1;
CMS FILEDEF DATAIN5 DISK TCF-ADJ DATA A1;
CMS FILEDEF DATAOUT1 DISK USMC MOVRSUP A (RECFM F LRECL 80 BLOCK 80 ;
CMS FILEDEF DATAOUT2 DISK USMC NONMSUP A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT3 DISK RESV-RET FILE A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT4 DISK ERROR FILE1 A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT5 DISK NON-MOVR SRTXOTYP (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT6 DISK MOVERS SRTXOTYP A (RECFM F LRECL 80 BLOCK 80;
*CMS FILEDEF DATAOUT7 DISK INV-SIZE DATA (RECFM F LRECL 80 BLOCK 80;
*CMS FILEDEF DATAOUT8 DISK ALPHA DATA A (RECFM F LRECL 80 BLOCK 80;
OPTIONS LINESIZE = 80;
OPTIONS MPRINT;
        /*
DEFINITION OF VARIABLES
      ADJTCF   -  ADJUSTMENT TO TOUR CONTROL FACTOR
      AMCC     -  ADVANCED MCC. PLANNED MCC AFTER NEXT ONE (ie TWO AHEAD)
      CCNAME   -  NAME OF COST CENTER
      COSTCTR  -  COST CENTER CODE
      DCTB     -  DATE THE CURRENT TOUR BEGAN
      DCTBIND  -  INDICATOR VARIABLE FOR DCTB
      EAS      -  END OF ACTIVE SERVICE. CONTRACT END DATE FOR RESERVISTS
      EASIND   -  INDICATOR VARIABLE FOR EAS
      EDA1     -  EXPECTED DATE OF ARRIVAL AT FMCC
      EDA1IND  -  INDICATOR VARIABLE FOR EDA1
      EDA2     -  EXPECTED DATE OF ARRIVAL AT AMCC
      EDA2IND  -  INDICATOR VARIABLE FOR EDA2
      EXP      -  EXPERIENCE  BINARY INDICATOR VARIABLE
      FMCC     -  FUTURE MCC. THE NEXT MCC FOR WHICH AN INDIV IS SLATED
```

```
GRADE    -  GRADE OR RANK OF INDIVIDUAL
GRD      -  GRADE OR RANK OF INDIVIDUAL
IDNUM    -  NUMBER USED TO IDENTIFY INDIVIDUALS IN THE SOLUTION.
ITD      -  INTENDED TRANSFER DATE: USUALLY ITD = DCTB + TCF
MOS      -  PRIMARY MOS
MOSNUM   -  MOS NUMBER: USED TO INDEX DIFFERENT MOS'S IN ARRAY WHICH
               USED IN THE MATCHING SUBROUTINE. (MOSGRD ARRAY A1)
MOSTYP   -  CHARACTER VARIABLE WHICH INDICATES THE FAMILY TO WHICH
               THE MOS BELONGS. (EG. NAFW, NARW, GCSS, GNDC, ETC.)
MOS2     -  SECONDARY MOS
MOS3     -  TERTIARY MOS
PMCC     -  PRESENT MCC (PRESENT LOCATION OF INDIVIDUAL)
SEX      -  SEX
TCF      -  TOUR CONTROL FACTOR. NO. OF MONTHS OF A "STANDARD" TOUR"
TDCTB    -  TEMPORARY VARIABLE TO STORE DCTB
TEAS     -  TEMPORARY VARIABLE TO STORE EAS
TEDA1    -  TEMPORARY VARIABLE TO STORE EDA1
TEDA2    -  TEMPORARY VARIABLE TO STORE EDA2
TITD     -  TEMPORARY VARIABLE TO STORE ITD
WINDOW   -  TIME PERIOD, MEASURED FROM TODAY, OVER WHICH
               FIXED OR FREE STATUS IS DETERMINED

   THE FIRST DATA SET IS A LIST OF ALL THE VARIABLES SORTED BY MOS.
LATER, WHEN THIS SORTED LIST IS READ INTO THE MOVSUP AND NMOVSUP DATA
SETS (TO BE OUTPUT AS CMS DATA FILES), EACH UNIQUE MOS IS ASSIGNED AN
MOS NUMBER (MOSNUM) WHICH WILL LATER BE USED IN THE MAIN SORT-AND-MATCH
FORTRAN SUBROUTINE AS AN INDEX TO A MOS-GRADE ARRAY.  THE MOS-GRADE ARRAY
WILL BE USED TO SPEED UP ACCESS TIME BY FINDING THE ENTRY POINT INTO
THE INVENTORY FILE (EITHER MOVSUP OR NMOVSUP) WHEN FINDING INDIVIDUALS
WHO MATCH EACH E1 CARD.
  */;

DATA _NULL_;
INFILE DATAIN1;
  INPUT GRADE $1-2 MOS $4-7 MOS2 $9-12 MOS3 $14-17 LDO $19 TEAS $23-26
  PMCC $28-30 TDCTB $32-35 TCF 37-38 TITD $40-43 FMCC $45-47 TEDA1
  $49-52 AMCC $54-56 TEDA2 $59-62 EXP $64 SEX $66 EASIND $24 DCTBIND $33
  ITDIND $41 EDA1IND $50 EDA2IND $60 CFT 21;
  IDNUM + 1;
   MNTHDAY = '01'         ;
   IF TEAS NE ' ' THEN EAS = TEAS || '28';
   IF TITD NE '   0' AND TITD NE '0000' THEN ITD = TITD || MNTHDAY;
   IF TDCTB NE '   0' THEN DCTB = TDCTB || MNTHDAY;
   IF TEDA1 NE '   0' AND TEDA1 NE '0000' THEN EDA1 = TEDA1 || MNTHDAY;
   IF TEDA2 NE '   0' THEN EDA2 = TEDA2 || MNTHDAY;
   IF EASIND EQ ' ' THEN EAS = ' ';
   IF DCTBIND EQ ' ' THEN DCTB = ' ';
   IF ITDIND EQ ' ' OR TITD EQ '0000' THEN ITD = ' ';
   IF EDA1IND EQ ' ' OR TEDA1 EQ '0000' THEN EDA1 = ' ';
   IF EDA2IND EQ ' ' THEN EDA2 = ' ';
* LDO IS ALREADY CODED. 1 = LDO. 2 = UNRESTRICTED;
  FILE DATAIN2;
    PUT GRADE $1-2 MOS $4-7 MOS2 $9-12 MOS3 $14-17 LDO $19 PMCC $21-23
    SEX $25 EXP $27 CFT 29 EAS $31-36 DCTB $38-43
    TCF 45-46 ITD $48-53 FMCC $55-57 EDA1 $60-65
    AMCC $66-68 EDA2 $69-74 IDNUM 75-79;

DATA INVENTRY ;
INFILE DATAIN2;
  INPUT GRADE $1-2 MOS $4-7 MOS2 $9-12 MOS3 $14-17 LDO $19 MCC $21-23
  SEX $25 EXP $27 CFT 29 @31 EAS YYMMDD6. @38 DCTB YYMMDD6.
  TCF 45-46 @48 ITD YYMMDD6. FMCC $55-57 @60 EDA1 YYMMDD6.
  AMCC $66-68 @69 EDA2 YYMMDD6. IDNUM 75-79;
***  THE FOLLOWING IF STATEMENTS CONVERT GRADE TO INTEGER VALUES FOR ***;
***  USE IN THE ENTRY POINT ARRAY AS AN INDEX.                      ***;
  IF GRADE EQ '03' THEN GRD = ' 3';
  ELSE IF GRADE EQ '02' THEN GRD = ' 2';
  ELSE IF GRADE EQ '04' THEN GRD = ' 4';
```

```
   ELSE IF GRADE EQ 'WO' THEN GRD = ' 1';
   ELSE IF GRADE EQ '05' THEN GRD = ' 5';
        IF SEX = 'M' THEN SEX = '1';
        ELSE IF SEX = 'F' THEN SEX ='2';
        ELSE IF SEX = '*' THEN SEX = '0';
PROC SORT DATA = INVENTRY;
        BY MOS GRD MOS2 MOS3;

    /*
     AT THIS POINT A SERIES OF "IF THEN, ELSE IF" STATEMENTS ARE INSERTED
TO ASSIGN TO EACH PMOS AN APPROPRIATE MOSTYP SUCH AS NAFW, NARW,GCSS,etc.
   */
DATA BCARDS;
    INFILE DATAIN3;
    INPUT MOS $1-4 MOSTYP $8-11;
    IF MOSTYP = 'AD  ' OR MOSTYP = 'BL  ' THEN DELETE;
PROC SORT DATA = BCARDS;
        BY MOS;
DATA FILEX;
    MERGE INVENTRY BCARDS;
        BY MOS;
    IF MOSTYP = 'NAFW' THEN OFFTYP = '10000000';
    ELSE IF MOSTYP = 'NAHE' THEN OFFTYP = '01000000';
    ELSE IF MOSTYP = 'NANF' THEN OFFTYP = '00100000';
    ELSE IF MOSTYP = 'GDCB' THEN OFFTYP = '00010000';
    ELSE IF MOSTYP = 'GDCS' THEN OFFTYP = '00001000';
    ELSE IF MOSTYP = 'GCSS' THEN OFFTYP = '00000100';
    ELSE IF MOSTYP = 'AGCS' THEN OFFTYP = '00000010';
    ELSE IF MOSTYP = 'AGSS' THEN OFFTYP = '00000001';
    ELSE IF MOSTYP = 'NOSG' THEN OFFTYP = '00000000';
*  ELSE DO;
*     FILE DATAOUT4;
*     PUT MOS 1-4 OFFTYP $5-12 MOSTYP $14-17;
*  END;

*****   read in TCF adjustment factor from TCF-ADJ DATA   *****;
* DEFINE SAS MACRO;
%MACRO MAC1;
    ELSE IF DCTB + ((TCF + &ADJTCF)*365.25/12) >= WINDOW THEN DO;
%MEND;
DATA DCHTCF;
    INFILE DATAIN5;
    INPUT CHGTCF 1-5;
*  RETAIN CHGTCF;
    CALL SYMPUT('ADJTCF',CHGTCF);
RUN;
* CLEAR OUT SOME UNNEEDED DATA SETS;
PROC DATASETS;
    DELETE INVENTRY BCARDS;
*     NOW THE COST CODE CENTERS ARE ADDED TO THE MCC'S WHICH          ;
* WILL APPEAR IN THE OUTPUT FILE.  THESE COST CODE CENTERS WILL BE USED;
* IN THE MATCHING ROUTINE TO REFERENCE AN ARRAY CONTAINING THE COSTS   ;
* ASSOCIATED WITH MOVING AN OFFICER OF SOME PARTICULAR RANK FROM HIS   ;
* PRESENT MCC (WHICH APPEAR ON THE "USMC MOVRSUP" OR "NMOVSUP" FILE)   ;
* TO HIS PROPOSED FUTURE MCC ("FMCC" - WHICH IS PULLED FROM THE FILES  ;
* MOVR-DEM INPUT AND NMOV-DEM INPUT FROM E1ASRE1A/B SAS.               ;
* BECAUSE SOME OF THE MCC'S ON THE PRESENT COST CODE CENTER LIST   ;
* ARE NOT YET PROPERLY MATCHED WITH A COST CODE CENTER, THESE WILL BE    ;
* ARBITRARILY ASSIGNED TO COST CODE CENTER NUMBER 29 - KANSAS CITY. ;
* ADD CCC'S;
DATA DD1;
    INFILE DATAIN4;
    INPUT @7 MCC $CHAR3. @11 COSTCTR $CHAR2. @14 CCNAME $CHAR10.;
    IF COSTCTR = ' 0' THEN DO;
       COSTCTR = '29';
       CCNAME = '*WARNING!*';
    END;
```

```
PROC SORT DATA = DD1;
    BY MCC;
PROC SORT DATA = FILEX;
    BY MCC;
DATA TOTSUP;
    MERGE DD1 FILEX;
    BY MCC;
    IF COSTCTR = ' ' THEN COSTCTR = '**';
PROC SORT DATA = TOTSUP;
    BY MOS GRD MOS2 MOS3;
*****  Determine window of the problem.  This is set to 1 year.  *****;
    /*
     IF THE INDIVIDUAL'S ROTATION DATE FALLS WITHIN ONE YEAR OF TODAY'S
     DATE (IE. WHENEVER THE JOB IS RUN), THEN HE IS CLASSIFIED AS A MOVER,
     AND HIS RECORD IS PLACED IN THE MOVSUP (MOVER SUPPLY) FILE. iF NOT,
     his RECORD IS PLACED IN THE NMOVSUP (NON-MOVER SUPPLY FILE).  FURTHER
     REFINEMENTS OF THIS JOB WILL PERMIT THE USER TO INPUT BOTH THE DATE
     FROM WHICH THE MOVERS ARE DETERMINED, AND THE NUMBER OF DAYS OR
     MONTHS FROM THAT DATE WITHIN WHICH A RECORD WILL BE ASSIGNED TO EACH
     FILE.  ALSO, THIS IS WHERE THOSE WHO ARE FIXED AT AN MCC WILL BE SO
     DESIGNATED USING A "LOGICAL IF" STATEMENT */;
DATA _NULL_;
      SET TOTSUP;
*    IDNUM + 1;
      IF MOS NE LAG(MOS) THEN MOSNUM + 1;
      TDAY = TODAY();
      WINDOW = TDAY + 365.25;

********************   CREATE FILE OF NON-MOVERS   ********************;
IF AMCC NE '    ' AND EDA2 >= WINDOW THEN DO;
      FILE DATAOUT2;
      FIX = 1;
      IF GRD = '   ' THEN DELETE;
      PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
          EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
          @65 EDA2 YYMMDD4. OFFTYP $70-77 COSTCTR $79-80;
    END;
ELSE IF FMCC NE '    ' AND EDA1 >= WINDOW THEN DO;
      FILE DATAOUT2;
      FIX = 1;
      IF GRD = '   ' THEN DELETE;
      PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
          EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
          @61 EDA1 YYMMDD4. OFFTYP $70-77 COSTCTR $79-80;
END;
ELSE IF ITD  >= WINDOW THEN DO;
      FILE DATAOUT2;
      FIX = 1;
      IF GRD = '   ' THEN DELETE;
* NOTE THAT ITD, EDA1, EDA2, AND MOVR ARE NOT NECESSARY AND WERE ONLY;
* INCLUDED IN THE OUTPUT TO FACILITATE DE-BUGGING.  THEY MAY BE REMOVED;
* WHEN NO LONGER NEEDED FOR CLARITY;
      PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
          EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
          @57 ITD YYMMDD4. OFFTYP $70-77 COSTCTR $79-80;
END;
      %MAC1;
      FILE DATAOUT2;
      FIX = 1;
      IF GRD = '   ' THEN DELETE;
      PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
          EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
          @50 DCTB YYMMDD4.TCF 55-56 OFFTYP $70-77 COSTCTR $79-80;
END;
*****   CREATE FILE OF POTENTIAL MOVERS AND REMOVE RETIREES/EAS'S  *****;
IF FIX NE 1 THEN DO;
     IF AMCC NE '    ' THEN DO;
```

```
        IF EDA2 <= WINDOW THEN DO;
         RESERVE = SUBSTR(AMCC,1,2);
         IF AMCC = 'W95' OR RESERVE EQ 'ZY' THEN DO;
          FILE DATAOUT3;
          IF GRD = '  ' THEN DELETE;
          PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           @41 DCTB YYMMDD4. TCF 46-48 @50 ITD YYMMDD4.
           @55 EDA1 YYMMDD4. @60 EDA2 YYMMDD4. AMCC $65-67 OFFTYP $70-77
           COSTCTR $79-80;
           DELETE;
         END;
        END;
       END;
      FILE DATAOUT1;
        FIX = 0;
        IF GRD = '  ' THEN DELETE;
* NOTE THAT ITD, EDA1, EDA2, AND MOVR ARE NOT NECESSARY AND WERE ONLY;
* INCLUDED IN THE OUTPUT TO FACILITATE DE-BUGGING.  THEY MAY BE REMOVED;
* WHEN NO LONGER NEEDED FOR CLARITY;
        PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           @41 DCTB YYMMDD4. TCF 46-48 @50 ITD YYMMDD4.
           @55 EDA1 YYMMDD4. @60 EDA2 YYMMDD4. AMCC $65-67 OFFTYP $70-77
           COSTCTR $79-80;
END;

   /*

      IN THIS NEXT SECTION, THE FILES CREATED ABOVE ARE SORTED BY MOS GRADE
      MCC, OR BY OFFTYP GRD AND MOS'S FOR USE AS INPUT FILES TO FORTRAN
      PROGRAMS WHICH WILL CREATE ADJACENCY LISTS WHICH WILL BE USED
      TO SPEED UP THE SORTING AND MATCHING PROCESS. THE SET MOVSUP
      IS ALREADY SORTED BY MOS AND GRADE  AND IS USED IN THE
      PRIMARY SORT AND MATCH ROUTINE.
         */;
DATA FILEW;
    INFILE DATAOUT1;
      INPUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           OFFTYP $70-77 COSTCTR $79-80;
PROC SORT DATA = FILEW;
    BY OFFTYP GRD MOS MOS2 MOS3;
DATA _NULL_;
      SET FILEW;
      IF GRD = '  ' OR MCC = '  ' THEN DELETE;
FILE DATAOUT6;
      PUT OFFTYP $1-8 GRD $10-11 MOS $13-16 MOS2 $18-21 MOS3 $23-26
      LDO $28 SEX $30 EXP $32 MCC $34-36 COSTCTR $38-39;
DATA FILEY;
    INFILE DATAOUT2;
      INPUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           OFFTYP $70-77 COSTCTR $79-80;
PROC SORT DATA = FILEY;
    BY OFFTYP GRD MOS MOS2 MOS3;
DATA _NULL_;
    SET FILEY;
    IF GRD = '  ' OR MCC = '  ' THEN DELETE;
FILE DATAOUT5;
      PUT OFFTYP $1-8 GRD $10-11 MOS $13-16 MOS2 $18-21 MOS3 $23-26
      LDO $28 SEX $30 EXP $32 MCC $34-36 COSTCTR $38-39;
```

APPENDIX D
ADJ-LIST FORTRAN

1. PROGRAM TO GENERATE POINTER ARRAY FOR MOVERS AND NON-MOVERS

```
**********************************************************************
*                                                                    *
*      *  *  *  *   PROGRAM NAME:  ADJ-LIST FORTRAN   *  *  *  *     *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 * * *   OVERVIEW AND PURPOSE   * * *               *
*                                                                    *
*    THIS PROGRAM GENERATES A POINTER ARRAY WHICH WILL BE INPUT TO   *
* ENTRY-PT FORTRAN FOR EXPANSION INTO AN ARRAY THAT WILL BE USED TO  *
* HELP IN THE MATCHING PROCESS.  WHEN THE PEOPLE ARE MATCHED TO      *
* BILLETS, IT WILL BE NECESSARY TO SEARCH THROUGH THE INVENTORY TO   *
* FIND ALL THOSE IN THE INVENTORY WHO FILL THE QUALIFICATIONS LISTED *
* ON THE E1 CARD THAT IS BEING EXAMINED.                             *
*      THIS PROGRAM GENERATES AN ENTRY POINT ARRAY FOR BOTH THE FIXED *
* AND THE FREE INVENTORIES WHICH WILL ENABLE THE MATCHING ROUTINE TO *
* SEARCH ONLY AMONG THOSE WITH THE PROPER MOS AND GRADE WHEN IT MUST *
* MATCH PEOPLE TO BILLETS.  THE ARRAYS POINT TO THE FIRST AND LAST   *
* OCCURRENCE OF A PARTICULAR MOS/GRADE COMBINATION IN THE FIXED AND  *
* FREE INVENTORIES.                                                  *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 ***   FILE DEFINITIONS   ***                       *
*                                                                    *
*    FILEDEF    FILE IDENTIFICATION              PURPOSE             *
*       8         USMC MOVRSUP          CONTAINS ALL MOVERS IN USMC  *
*       9         ADJ-MOVR ARRAY        HOLDS MOVER  POINTER ARRAY   *
*      10         USMC NONMSUP          CONTAINS ALL USMC NON_MOVERS *
*      11         ADJ-NMOV ARRAY        HOLDS NON_MOVER POINTER ARRAY *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                ***   DEFINITION OF TERMS   ***                     *
*                                                                    *
*   BEGINX    STARTING POINT OF THE CURRENT MOS/GRADE COMBINATION    *
*   ENDAT     ENDING POINT ON LIST OF CURRENT MOS/GRADE COMBINATION  *
*   GRD       GRADE OR RANK                                          *
*   MOS       THE MOS CURRENTLY BEING LOOKED AT                      *
*   MOSNUM    THE NUMBER OF THE CURRENT MOS IN THE LIST              *
*   OLDGRD    THE PREVIOUS GRADE THAT WAS LOOKED AT IN THE LIST      *
*   OLDMOS    THE PREVIOUS MOS LOOKED AT                             *
*   POSIT     COUNTER TO MARK POSITION IN THE OVERALL LIST           *
*   SEARCH    ARRAY VARIABLE NAME CONTAINING START & ENDING POINTS   *
*                                                                    *
**********************************************************************
*     DEBUG SUBCHK
*     ENDDEBUG
      INTEGER POSIT, BEGINX, ENDAT, SEARCH, GRD, OLDGRD,MOSNUM
      CHARACTER*4 MOS, OLDMOS*4
      DIMENSION SEARCH(180,6,2)
      POSIT = 0
***********     INITIALIZE ARRAY    ********
      DO 10 I = 1,180
         DO 10 J = 1,6
            DO 10 K = 1,2
               SEARCH(I,J,K) = 0
```

```
10     CONTINUE
15     READ (11,101, END = 998) MOS, GRD
101    FORMAT(A4,2X,I1)
         POSIT = POSIT + 1
          IF (POSIT .EQ. 1) THEN
              OLDGRD = GRD
              OLDMOS = MOS
              MOSNUM = 1
              BEGINX = 1
          ELSE IF ((MOS .EQ. OLDMOS) .AND. (GRD .NE. OLDGRD)) THEN
              ENDAT = POSIT - 1
              SEARCH(MOSNUM,OLDGRD,1) = BEGINX
              SEARCH(MOSNUM,OLDGRD,2) = ENDAT
              BEGINX = POSIT
              WRITE(12,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
      C      SEARCH(MOSNUM,OLDGRD,2)
             OLDGRD = GRD
         ELSE IF ((MOS .NE. OLDMOS) .AND. (GRD .EQ. OLDGRD)) THEN
              ENDAT = POSIT - 1
              SEARCH(MOSNUM,OLDGRD,1) = BEGINX
              SEARCH(MOSNUM,OLDGRD,2) = ENDAT
              WRITE(12,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
      C      SEARCH(MOSNUM,OLDGRD,2)
              MOSNUM= MOSNUM + 1
              BEGINX = POSIT
              OLDMOS = MOS
          ELSE IF ((MOS .NE. OLDMOS) .AND. (GRD .NE. OLDGRD)) THEN
              ENDAT = POSIT - 1
              SEARCH(MOSNUM,OLDGRD,1) = BEGINX
              SEARCH(MOSNUM,OLDGRD,2) = ENDAT
              WRITE(12,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
      C      SEARCH(MOSNUM,OLDGRD,2)
              MOSNUM = MOSNUM + 1
              BEGINX = POSIT
              OLDMOS = MOS
              OLDGRD = GRD
102    FORMAT(I3,1X,A4,1X,I1,1X,I6,I6)
         ENDIF
         GO TO 15
998          ENDAT = POSIT
              SEARCH(MOSNUM,OLDGRD,1) = BEGINX
              SEARCH(MOSNUM,OLDGRD,2) = ENDAT
              WRITE(12,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
      C      SEARCH(MOSNUM,OLDGRD,2)
*  NOW WE HAVE FINISHED GENERATING THE ENTRY POINT ARRAY FOR USMC MOVSUP
* WE WILL NOW RE-INITIALIZE THE SAME VARIABLES AND PERFORM A SIMILAR
* FUNCTION FOR THE FILE USMC NMOVSUP (THE NON-MOVERS)
*****************    RE-INITIALIZE POSITION COUNTER     ****************
         POSIT = 0
***********************     RE-INITIALIZE ARRAY    ********************

         DO 20 I = 1,180
             DO 20 J = 1,6
                 DO 20 K = 1,2
                     SEARCH(I,J,K) = 0
20     CONTINUE
25     READ (13,101, END = 999) MOS, GRD
         POSIT = POSIT + 1
          IF (POSIT .EQ. 1) THEN
              OLDGRD = GRD
              OLDMOS = MOS
              MOSNUM = 1
              BEGINX = 1
          ELSE IF ((MOS .EQ. OLDMOS) .AND. (GRD .NE. OLDGRD)) THEN
              ENDAT = POSIT - 1
              SEARCH(MOSNUM,OLDGRD,1) = BEGINX
              SEARCH(MOSNUM,OLDGRD,2) = ENDAT
              BEGINX = POSIT
```

```
            WRITE(14,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
     C    SEARCH(MOSNUM,OLDGRD,2)
            OLDGRD = GRD
       ELSE IF ((MOS .NE. OLDMOS) .AND. (GRD .EQ. OLDGRD)) THEN
            ENDAT = POSIT - 1
            SEARCH(MOSNUM,OLDGRD,1) = BEGINX
            SEARCH(MOSNUM,OLDGRD,2) = ENDAT
            WRITE(14,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
     C    SEARCH(MOSNUM,OLDGRD,2)
            MOSNUM= MOSNUM + 1
            BEGINX = POSIT
            OLDMOS = MOS
        ELSE IF ((MOS .NE. OLDMOS) .AND. (GRD .NE. OLDGRD)) THEN
            ENDAT = POSIT - 1
            SEARCH(MOSNUM,OLDGRD,1) = BEGINX
            SEARCH(MOSNUM,OLDGRD,2) = ENDAT
            WRITE(14,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
     C    SEARCH(MOSNUM,OLDGRD,2)
            MOSNUM = MOSNUM + 1
            BEGINX = POSIT
            OLDMOS = MOS
            OLDGRD = GRD
       ENDIF
       GO TO 25
999       ENDAT = POSIT
            SEARCH(MOSNUM,OLDGRD,1) = BEGINX
            SEARCH(MOSNUM,OLDGRD,2) = ENDAT
            WRITE(14,102) MOSNUM, OLDMOS, OLDGRD, SEARCH(MOSNUM,OLDGRD,1),
     C    SEARCH(MOSNUM,OLDGRD,2)
       CLOSE(11)
       CLOSE(12)
       CLOSE(13)
       CLOSE(14)
       STOP
       END
```

APPENDIX E
ENTRY-PT FORTRAN

1. PROGRAM TO COMPLETE THE GENERATION OF THE POINTER ARRAYS

```
************************************************************************
*                                                                      *
*       *  *  *  *   PROGRAM NAME: ENTRY-PT FORTRAN    *  *  *  *      *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                 * * *   OVERVIEW AND PURPOSE   * * *                 *
*                                                                      *
*     THIS PROGRAM EXPANDS THE BASIC ENTRY POINT ARRAY GENERATED IN    *
* ADJ-LIST FORTRAN INTO A FORMAT THAT CAN BE USED BY THE MATCHING      *
* PROGRAM.  THE ADJACENCY LIST THAT IS READ IN FROM THE ADJ-LIST OMITS *
* ALL MOS/GRADE COMBINATIONS THAT DON'T APPEAR IN THE INVENTORY, BUT   *
* IN ORDER TO BE A GENERAL PURPOSE MATCHING PROGRAM THE MATCHING       *
* ROUTINE MUST LOOK FOR ALL MOS/GRADE COMBINATIONS.  IN ORDER TO ALLOW *
* IT TO USE THE LIST, THE POINTER ARRAY MUST INDICATE ALL COMBINATIONS,*
* EVEN THOSE WHICH DO NOT APPEAR IN THE INVENTORY.  THIS PROGR'M TAKES *
* THE COMPACT ENTRY POINT ARRAY AND EXPANDS IT INTO A FORMAT THAT      *
* INCLUDES ALL MOS/GRADE COMBINATIONS.                                 *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                  ***   FILE DEFINITIONS   ***                        *
*                                                                      *
*   FILEDEF    FILE IDENTIFICATION              PURPOSE                *
*      9         ADJ-MOVR ARRAY        HOLDS MOVER  POINTER ARRAY      *
*     11         ADJ-NMOV ARRAY        HOLDS NON_MOVER POINTER ARRAY   *
*     12         MOVR-EP ARRAY         EXPANDED MOVER POINTER ARRAY    *
*     13         NMOVREP ARRAY         EXPANDED NON-MOVER ARRAY        *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                 ***   DEFINITION OF TERMS   ***                      *
*                                                                      *
*  BEGINX    STARTING POINT OF THE CURRENT MOS/GRADE COMBINATION       *
*  ENDAT     ENDING POINT ON LIST OF CURRENT MOS/GRADE COMBINATION     *
*  GRD       GRADE OR RANK                                             *
*  MOS       THE MOS CURRENTLY BEING LOOKED AT                         *
*  MOSNUM    THE NUMBER OF THE CURRENT MOS IN THE LIST                 *
*  OLDGRD    THE PREVIOUS GRADE THAT WAS LOOKED AT IN THE LIST         *
*  OLDMOS    THE PREVIOUS MOS LOOKED AT                                *
*  POSIT     COUNTER TO MARK POSITION IN THE OVERALL LIST              *
*                                                                      *
************************************************************************
       INTEGER BEGINX, ENDAT, GRD,MOSNUM,OLDNUM,ZERO,M,OLDM
       CHARACTER*4 MOS,OLDMOS
       ZERO = 0
       M = 1

1     READ (12,103, END = 998) MOSNUM, MOS, GRD, STARTX, ENDAT
103   FORMAT(I3,1X,A4,1X,I1,1X,I6,I6)
       IF ((MOSNUM .EQ.1) .AND. (M .EQ. 1)) THEN
          OLDMOS = MOS
          OLDNUM = MOSNUM
       ENDIF

2     IF ((M .NE. GRD) .AND. (MOS .EQ. OLDMOS)) THEN
            WRITE(13,103) MOSNUM, MOS,M,ZERO,ZERO
            OLDM = M
            M = M+1
*         IF (M .EQ. 6) M = 1
            GO TO 2
       ELSE IF ((M .EQ. GRD) .AND. (MOS .EQ. OLDMOS)) THEN
            WRITE(13,103) MOSNUM, MOS, GRD,STARTX,ENDAT
            OLDM = M
            M = M+1
            IF (M .EQ. 6) M = 1
            GO TO 1
       ELSE IF ((M .EQ. GRD) .AND. (MOS .NE. OLDMOS)) THEN
           IF (M .EQ. 1) THEN
              OLDMOS = MOS
              OLDNUM = MOSNUM
              WRITE(13,103) MOSNUM, MOS, GRD,STARTX,ENDAT
              OLDM = M
              M = M+1
              GO TO 1
           ELSE IF (M .NE. 1) THEN
              IF (OLDM .EQ. 5) THEN
                 WRITE(13,103) MOSNUM, MOS, GRD,STARTX,ENDAT
                 OLDM = M
                 M = M+1
                 IF (M .EQ. 6) M = 1
                 GO TO 2
              ELSE IF (OLDM .NE. 5) THEN
                 WRITE(13,103) OLDNUM,OLDMOS,M,ZERO,ZERO
                 OLDM = M
                 M = M+1
                 IF (M .EQ. 6) M = 1
                 GO TO 2
              ENDIF
           ENDIF
       ELSE IF ((M .NE. GRD) .AND. (MOS .NE. OLDMOS)) THEN
           IF (M .EQ.1) THEN
              WRITE(13,103) MOSNUM,MOS,M,ZERO,ZERO
              OLDMOS = MOS
              OLDNUM = MOSNUM
              OLDM = M
              M = M+1
              GOTO 2
           ELSE IF (M .NE. 1) THEN
              WRITE(13,103) OLDNUM,OLDMOS,M,ZERO,ZERO
              OLDM = M
              M = M+1
              IF (M .EQ. 6) M = 1
              GOTO 2
           ENDIF
       ENDIF
998   DO 20 J = M,5
           WRITE(13,103) MOSNUM,MOS,J,ZERO,ZERO
20    CONTINUE
*   ONCE AGAIN WE WILL TEMPT FATE AND ATTEMPT TO FILL OUT THE ENTRY
* ARRAY FOR ADJ-NMOV ARRAY A1 IMMEDIATELY AFTER DOING IT FOR ADJ-MOVR
* ARRAY A1 AND USING THE SAME VARIABLES.
******************     RE-INITIALIZE VARIABLES     *********************
       ZERO = 0
       M = 1
*************     BEGIN TO READ FROM ADJ-NMOV ARRAY     ***************
3     READ (14,103, END = 999) MOSNUM, MOS, GRD, STARTX, ENDAT
       IF ((MOSNUM .EQ.1) .AND. (M .EQ. 1)) THEN
           OLDMOS = MOS
           OLDNUM = MOSNUM
       ENDIF
4     IF ((M .NE. GRD) .AND. (MOS .EQ. OLDMOS)) THEN
            WRITE(15,103) MOSNUM, MOS,M,ZERO,ZERO
            OLDM = M
            M = M+1
```

```
             GO TO 4
         ELSE IF ((M .EQ. GRD) .AND. (MOS .EQ. OLDMOS)) THEN
             WRITE(15,103) MOSNUM, MOS, GRD,STARTX,ENDAT
             OLDM = M
             M = M+1
             IF (M .EQ. 6) M = 1
             GO TO 3
         ELSE IF ((M .EQ. GRD) .AND. (MOS .NE. OLDMOS)) THEN
            IF (M .EQ. 1) THEN
               OLDMOS = MOS
               OLDNUM = MOSNUM
               WRITE(15,103) MOSNUM, MOS, GRD,STARTX,ENDAT
               OLDM = M
               M = M+1
               GO TO 3
            ELSE IF (M .NE. 1) THEN
               IF (OLDM .EQ. 5) THEN
                  WRITE(15,103) MOSNUM, MOS, GRD,STARTX,ENDAT
                  OLDM = M
                  M = M+1
                  IF (M .EQ. 6) M = 1
                  GO TO 4
               ELSE IF (OLDM .NE. 5) THEN
                  WRITE(15,103) OLDNUM,OLDMOS,M,ZERO,ZERO
                  OLDM = M
                  M = M+1
                  IF (M .EQ. 6) M = 1
                  GO TO 4
               ENDIF
            ENDIF
         ELSE IF ((M .NE. GRD) .AND. (MOS .NE. OLDMOS)) THEN
            IF (M .EQ.1) THEN
               WRITE(15,103) MOSNUM,MOS,M,ZERO,ZERO
               OLDMOS = MOS
               OLDNUM = MOSNUM
               OLDM = M
               M = M+1
               GOTO 4
            ELSE IF (M .NE. 1) THEN
               WRITE(15,103) OLDNUM,OLDMOS,M,ZERO,ZERO
               OLDM = M
               M = M+1
               IF (M .EQ. 6) M = 1
               GOTO 4
            ENDIF
         ENDIF
999     DO 30 J = M,5
            WRITE(15,103) MOSNUM,MOS,J,ZERO,ZERO
30      CONTINUE
         CLOSE(12)
         CLOSE(13)
         CLOSE(14)
         CLOSE(15)
         STOP
         END
```

APPENDIX F
ROLINV SAS

1. PROGRAM TO CONVERT THE NON-MOVER INVENTORY TO ASR FORMAT.

```
************************************************************************
*                                                                      *
*       *  *  *  *   PROGRAM NAME: ROLINV SAS          *  *  *  *      *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                  * * *   OVERVIEW AND PURPOSE   * * *                *
*                                                                      *
*      THIS PROGRAM TAKES THE NON-MOVER FILE AND ROLLS UP THE TOTALS   *
* OF NON-MOVERS IN EACH MOS AND GRADE.  IT THEN PUTS IT INTO THE SAME  *
* FORMAT AS THE ASR FILE.  THE OUTPUT FROM THIS FILE WILL BE USED IN   *
* THE PROGRAM INVENTRY1 SAS WHICH SUBTRACTS THE NON-MOVERS FROM THE    *
* ASR IN AN EFFORT TO REDUCE THE NUMBER OF UNNECESSARY MATCHES MADE    *
* BY THE MATCHING ROUITNE.                                             *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
**********************      FILE DEFINITION     ***********************
CMS FILEDEF FONE DISK USMC NONMSUP;
CMS FILEDEF FTWO DISK ROLNM INV (RECFM F LRECL 80 BLOCK 80;
************************************************************************

*THIS PROGRAM ROLLS THE FIXED INV INTO A MOS/GD/MCC/CNT FORMAT;
*SO THAT THE ASR CAN BE REDUCED BY THIS AMOUNT;
*INPUT INVENTORY;
DATA DONE (KEEP = MOS GD MCC MGM CNT);
  IF _N_ = 1 THEN CNT = 1;
  INFILE FONE;
  INPUT MOS 1-4 GD 7 MCC $ 19-21;
  TMP = MOS || GD;
  MGMA = TMP || MCC;
  IF MGMA = LAG(MGMA) THEN CNT +1;
  ELSE DO;
    MGM = LAG(MGMA);
    OUTPUT;
    CNT = 1;
  END;
PROC SORT;
  BY MGM;
DATA _NULL_;
  SET DONE;
  FILE FTWO;
  PUT MOS 1-4 GD 7 MCC 9-11 CNT 14-16;
```

APPENDIX G
XPASR SAS

1. PROGRAM TO READ IN THE USMC ASR DATA FILE.

```
**********************************************************************
*                                                                    *
*      *  *  *  *   PROGRAM NAME: XPASR SAS          *  *  *  *      *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 * * *   OVERVIEW AND PURPOSE   * * *               *
*                                                                    *
*       THIS PROGRAM READS THE USMC ASR FILE INTO THE SYSTEM.  IN THE  *
*   PROTOTYPE, THE USMC ASR TAPE IS CALLED RAW ASR.                   *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 ***   DEFINITION OF TERMS   ***                    *
*                                                                    *
*    BMOS      BILLET MILITARY OCCUPATIONAL SPECIALTY  (BMOS)         *
*    CAPT      THE NUMBER OF CAPTAINS AUTHORIZED AT THE INDICATED MCC  *
*    COL       THE NUMBER OF COLONELS AUTHORIZED AT THE INDICATED MCC  *
*    GD        GRADE INDICATOR                                       *
*    LT        THE NUMBER OF LIEUTENANTS AUTHORIZED AT THE MCC        *
*    LTCOL     THE NUMBER OF LIEUTENANT COLONELS AUTHORIZED AT THE MCC *
*    MAJ       THE NUMBER OF MAJORS AUTHORIZED AT THE MCC             *
*    MCC       MONITORED COMMAND CODE. THE LOCATION BEING INDICATED.  *
*    WO        THE NUMBER OF WARRANT OFFICERS AUTHORIZED AT THE MCC   *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *     FILE DEFINITIONS      *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK RAW ASR A;
CMS FILEDEF FTWO DISK NEW ASR A (RECFM F LRECL 80 BLOCK 80;
**********************************************************************


DATA DONE;
    INFILE FONE;
    FILE FTWO;
    INPUT BMOS 1-4 MCC $ 6-8 COL 24-28 LTCOL 30-34 MAJ 36-40
           CAPT 42-46 LT 48-52 WO 54-58;
    DROP COL LTCOL MAJ CAPT LT WO;
    IF COL NE 0 THEN DO;
      GD = 6;
      NUM = COL;
      PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
    END;
    IF LTCOL NE 0 THEN DO;
      GD = 5;
      NUM = LTCOL;
      PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
    END;
    IF MAJ NE 0 THEN DO;
      GD = 4;
      NUM = MAJ;
      PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
    END;
    IF CAPT NE 0 THEN DO;
      GD = 3;
      NUM = CAPT;
      PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
    END;
    IF LT NE 0 THEN DO;
      GD = 2;
    NUM = LT;
    PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
  END;
  IF WO NE 0 THEN DO;
    GD = WO;
    NUM = WO;
    PUT BMOS 1-4 MCC 8-10 GD 14-15 NUM 18-21;
  END;
```

APPENDIX H
C1ASR SAS

1. PROGRAM TO UPDATE THE ASR

```
**************************************************************************
*                                                                        *
*       *  *  *  *     PROGRAM NAME: C1ASR SAS         *  *  *  *        *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                        *
*                  * * *   OVERVIEW AND PURPOSE   * * *                  *
*                                                                        *
*   THIS PROGRAM UPDATES THE ASR USING THE C1ASR FILE WHICH CONTAINS     *
* ANY CHANGES SINCE THE LAST SEMI-ANNUAL ASR UPDATE.  CHANGES MAY BE      *
* ADDITIONS, DELETIONS, OR RESETTING THE AUTHORIZATION TO A SPECIFIC     *
* VALUE.  ONE POSSIBLE ENHANCEMENT OF THE SYSTEM WOULD BE TO INCLUDE      *
* THE CAPABILITY OF ADJUSTING THE C1ASR FILE TO ADD OR DELETE ENTIRE     *
* UNITS AT A CERTAIN MCC, OR TO CHANGE THE AUTHORIZATION OF A            *
* PARTICULAR OCC FIELD OR MOS ACROSS ALL MCC'S.                          *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                        *
*                  ***   FILE DEFINITIONS   ***                          *
*                                                                        *
*    FILEDEF    FILE IDENTIFICATION            PURPOSE                   *
*       8         USMC MOVRSUP          CONTAINS ALL MOVERS IN USMC      *
*       9         ADJ-MOVR ARRAY        HOLDS MOVER  POINTER ARRAY       *
*      10         USMC NONMSUP          CONTAINS ALL USMC NON_MOVERS     *
*      11         ADJ-NMOV ARRAY        HOLDS NON_MOVER POINTER ARRAY    *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                        *
*                  ***   DEFINITION OF TERMS   ***                       *
*                                                                        *
*   ACT       THE ACTION TO BE TAKEN; ADD, SUBTRACT, OR SET EQUAL TO.    *
*   ADJ       THE ADJUSTMENT UP OR DOWN, OR THE AUTH VALUE TO BE SET     *
*   AMT       THE AMOUNT OF THE ACTION (ACT) TO BE TAKEN                 *
*   GD        GRADE OR RANK                                              *
*   MCC       MONITORED COMMAND CODE (IE. THE PRESENT LOCATION)          *
*   MOS       THE MOS CURRENTLY BEING LOOKED AT                          *
*   NUM       THE PRESENT AUTHORIZATION                                  *
*                                                                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *    FILE DEFINITIONS      *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK WKC1 CRD A;
CMS FILEDEF FTWO DISK NEW ASR A;
CMS FILEDEF FTHREE DISK WORK ASR (RECFM F LRECL 80 BLOCK 80;
**************************************************************************

*READ IN THE C1 CARDS AND PUT IN SINGLE LINE FORMAT;
DATA DONE;
  INFILE FONE;
  INPUT MOS $ MCC $ GD ADJ $ @@;
  ACT = SUBSTR(ADJ,1,1);
  AMT = SUBSTR(ADJ,2,4);
  DROP ADJ;
*READ IN THE ASR;
DATA DTWO;
  INFILE FTWO;
  INPUT MOS $ MCC $ GD NUM ;
PROC SORT DATA = DONE;
  BY MOS MCC GD;
PROC SORT DATA = DTWO;
  BY MOS MCC GD;
```

```
DATA DTHREE;
   MERGE DTWO DONE;
   BY MOS MCC GD;
DATA DFOUR;
   SET DTHREE;
   IF ACT = 'E' THEN NUM = 0 + AMT;
   IF ACT = 'A' THEN NUM = NUM + AMT;
   IF ACT = 'S' THEN NUM = NUM- AMT;
DATA _NULL_;
   SET DFOUR;
   IF NUM = '.' THEN DELETE;
   IF NUM LT 0 THEN NUM = 0;
* OUTPUT THE NEWLY CALCULATED VALUES TO THE ASR
   FILE FTHREE;
   PUT MOS 1-4 MCC 7-9 GD 12 NUM 15-19;
```

APPENDIX I
INVNTRY1 SAS

1. PROGRAM TO REDUCE THE ASR BY THE NON-MOVERS

```
**********************************************************************
*                                                                    *
*     *  *  *  *   PROGRAM NAME: INVNTRY1 SAS        *  *  *  *      *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                  * * *  OVERVIEW AND PURPOSE   * * *               *
*                                                                    *
*    IN AN EFFORT TO REDUCE THE NUMBER OF UNNECESSARY ARCS GENERATED *
* BY THE MATCHING ROUTINE, THOSE BILLETS WHICH ARE OCCUPIED BY NON-  *
* MOVERS WILL BE SUBTRACTED FROM THE TOTAL DEMAND TO WHICH THE MOVERS *
* WILL BE MATCHED.  THIS PROGRAM IS THE FIRST OF THREE WHICH PERFORM *
* THAT TASK.  IN INVNTRY1 SAS, THE NON-MOVERS ARE MATCHED, AS BEST AS *
* CAN BE DONE WITHOUT USING THE E1 AND E2 CARDS, TO THE ASR.  THE GOAL *
* IS TO FIND OUT EXACTLY WHICH DEMANDS THE NON-MOVERS ARE FILLING.   *
*                                                                    *
*     THIS PROGRAM TAKES THE ROLLED UP INVENTORY OF NON-MOVERS AND   *
* REDUCES THE ASR BY THAT AMOUNT IN PREPARATION FOR PROCESSING BY THE *
* FORTRAN PROGRAM "EXCESS" WHICH FINDS THE EXCESSES AND THE FREE ASR *
*                                                                    *
*    THERE WILL NOT BE A PERFECT MATCH OF PEOPLE TO ASR AUTHORIZATIONS *
* SINCE THERE ARE MULTIPLE ACCEPTABLE SUBSTITUTIONS FOR MOST BILLETS. *
* IDEALLY, ONE SHOULD TAKE ALL NON-MATCHES (WHETHER AN UNFILLED BILLET *
* OR AN APPARENT EXCESS) AND CHECK AGAINST THE E-CARDS. IF THERE WERE *
* A PERSON WITH AN MOS-GRADE COMBINATION THAT WAS NOT AUTHORIZED AT A *
* PARTICULAR MCC ONE SHOULD CHECK THE SUBSTITUTIONS FOR THE OTHER    *
* BILLETS AT THAT MCC WHICH ARE AUTHORIZED TO SEE IF THE INDIVIDUAL'S *
* MOS AND GRADE FELL INTO ONE OF THE ACCEPTABLE SUBSTITUTION CRITERIA. *
* THE SEARCH COULD BE CONDUCTED THROUGH THE E-CARDS, BY ONLY LOOKING *
* AT BILLETS AT THAT MCC FOR WHICH THERE WAS AN AUTHORIZATION, BUT   *
* WHICH WERE NOT BEING FILLED.  TO PERFORM A PERFECTLY CORRECT CHECK *
* ON THE NUMBER OF EXCESSES, ONE WOULD NEED TO DO AN EXHAUSTIVE SEARCH *
* THROUGH ALL OF THE E-CARDS, SIMILAR TO THAT WHICH IS DONE IN THE   *
* MATCHING PROCESS. ADDITIONALLY, ONE WOULD HAVE TO DETERMINE WHICH OF *
* THOSE WHO ARE CURRENTLY HOLDING BILLETS FOR WHICH THEY DO NOT FIT  *
* SUBSTITUTIONS WERE "FIXED" THERE BY THE MONITOR, IN WHICH CASE, THE *
* INDIVIDUAL MAY NOT NECESSARILY BE EXCESS.  THIS WOULD, OF COURSE,  *
* INCLUDE CHECKING SECONDARY MOS'S AND IN MANY CASES ALLOWING GRADE  *
* SUBSTITUTIONS.  SUCH A CHECKER MIGHT BE BUILT INTO THE MODEL LATER, *
* BUT FOR THE PRESENT, WE WILL RESORT TO THE USE OF SOME HEURISTIC   *
* RULES TO REDUCE THE NUMBER OF INDIVIDUALS INCORRECTLY DESIGNATED AS *
* EXCESS.  THE USE OF A HEURISTIC IS JUSTIFIED IN LIGHT OF THE ACTUAL *
* EXCESS RECONCILIATION PROCESS WHICH INVOLVES PERMITTING REASONABLE *
* DEVIATIONS FROM THE SUBSTITUTION LIST CONTAINED IN THE E1 CARDS.   *
* AFTER ALL, THE PURPOSE OF THIS PORTION OF THE PROGRAM IS TO        *
* DETERMINE WHICH MCC'S ARE CONSIDERED BY CMC TO HAVE EXCESS PERSONNEL *
* BASED ON ACTUAL ASSIGNMENTS, NOT TO DETERMINE WHICH ASSIGNMENTS    *
* WOULD THEORETICALLY MATCH THE E-CARDS MOST PERFECTLY.  THAT WILL   *
* COME LATER.                                                        *
*                                                                    *
*    THE FOLLOWING DEPARTURES FROM A PERFECT MATCH BETWEEN THE ASR AND *
* THE INVENTORY WILL BE PERMITTED:                                   *
*                                                                    *
*       1.   GRADE SUBSTITUTIONS TO PLUS OR MINUS ONE GRADE, EXCEPT IN *
* THE CASE OF O5'S.                                                  *
*       2.   ANYONE WITH A 75XX AND THE APPROPRIATE GRADE MAY FILL   *
* A 9912 BILLET.                                                     *
*       3.   3060 OR 3070 BILLETS MAY BE PERSONS OF EITHER MOS.      *
*       4.   SINCE SECONDARY MOS'S WERE NOT CARRIED FORWARD FROM THE *
* INVENTORY TAPE, WE WILL MAKE SOME ASSUMPTIONS CONCERNING BILLETS     *
* NORMALLY REQUIRING CERTAIN SECONDARY MOS'S.  ANY 75XX OFFICER OR THE *
* APPROPRIATE GRADE MAY FILL 7596 OR 9958 BILLETS.                     *
*         5.   ONE IMPORTANT ADDITIONAL ENHANCEMENT WHICH COULD BE MADE *
* WITHOUT REFERENCE TO THE E-CARDS IS THE INCLUSION OF INDIVIDUALS'    *
* FUTURE MOS'S.  THEY DO NOT APPEAR ON THE "WORKING INVENTRY" FILE AT  *
* PRESENT, BUT COULD BE PULLED OFF WITHOUT TOO MUCH TROUBLE.           *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                 ***   DEFINITION OF TERMS   ***                      *
*                                                                      *
*   CNT       THE NUMBER OF PEOPLE ACTUALLY ON HAND AT THE MCC         *
*   FIRST2    THE FIRST 2 NUMBERS IN THE INDIV'S MOS (IE. OCC FIELD)   *
*   GD        GRADE OR RANK                                            *
*   MOS       THE MOS CURRENTLY BEING LOOKED AT                        *
*   MCC       MONITORED COMMAND CODE (IE. LOCATION)                    *
*   NUM       THE NUMBER OF PEOPLE AUTHORIZED IN THAT GRADE/MOS        *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *   FILE DEFINITION   *  *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK ROLNM INV A;
CMS FILEDEF FTWO DISK WORK ASR A;
CMS FILEDEF FFOUR DISK ASR-INV TOT-MRGE (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF FFIVE DISK NO-ASR CARD (RECFM F LRECL 80 BLOCK 80;
***********************************************************************
*INPUT INVENTORY;
DATA DONE;
   INFILE FONE;
   INPUT MOS 1-4 FIRST2 1-2 GD 7 MCC $ 9-11 CNT 14-16;
*INPUT ASR;
DATA DTWO;
   INFILE FTWO;
   INPUT MOS 1-4 FIRST2 1-2 MCC $ 7-9 GD 12 NUM 17-19;
PROC SORT DATA = DONE;
  BY MCC MOS GD;
PROC SORT DATA = DTWO;
  BY MCC MOS GD;
DATA DTHREE;
   MERGE DTWO DONE;
   BY MCC MOS GD;
* IF NUM IS MISSING THEN THERE IS NO ASR CARD FOR THAT MOS-GD-MCC;
* OR IF THE NUMBER ON HAND EXCEEDS AUTHORIZED STRENGTH, THERE ARE EXCESS;
   IF NUM = '.' THEN NUM = 0;
   IF CNT = '.' THEN CNT = 0;

   IF NUM = 0 OR NUM-CNT LE 0 THEN DO;
         IF NUM - CNT LE 0 THEN CNT = CNT - NUM;
         NUM = 0;
         FILE FFOUR;
         PUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16;
         FILE FFIVE;
         PUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16 FIRST2 18-19;
   END;
* IF CNT IS MISSING THEN NO ONE HAS FILLED ANY OF THAT REQUIREMENT YET;
   ELSE IF CNT = 0 THEN DO;
         FILE FFOUR;
         PUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16;
     END;
* IF THE NUMBER OF PEOPLE ON HAND (CNT) IS POSITIVE AND THE AUTHORIZED ;
* STRENGTH (NUM) IS GREATER THAN THE NUMBER ON HAND, THEN THERE IS SOME;
* EXTRA CAPACITY AT THAT BILLET WHICH EQUALS NUM - CNT;
* WE CAN TRANSFORM THIS TO A SIMPLE (TOTALLY) UNUSED CAPACITY AND SET  ;
* THE NUMBER ON HAND (CNT) TO ZERO;
   ELSE DO;
         NUM = NUM - CNT;
         CNT = 0;
         FILE FFOUR;
         PUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16;
   END;
```

APPENDIX J
ADJ-LIST SAS

1. PROGRAM TO GENERATE POINTER ARRAY FOR MAKING FREE ASR

```
**********************************************************************
*                                                                    *
*      * * * *   PROGRAM NAME: ADJ-LIST SAS        * * * *           *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 * * *   OVERVIEW AND PURPOSE   * * *               *
*                                                                    *
*   THIS PROGRAM GENERATES THE ADJACENCY LIST USED IN THE MAKING OF  *
* THE FREE ASR WHICH REMOVES NON-MOVERS FROM THE DEMAND.             *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*                 ***   DEFINITION OF TERMS   ***                    *
*                                                                    *
*   CNT        THE ACTUAL NUMBER OF PEOPLE ON HAND                   *
*   GD         GRADE OR RANK                                         *
*   MCC        MONITORED COMMAND CODE (IE. LOCATION)                 *
*   MCCNUM     THE NUMBER OF THE CURRENT MCC IN THE LIST             *
*   NUM        THE AUTHORIZATION FOR THE GRADE/MCC AT THE MCC        *
*   STARTAT    INDICATES LOCATION IN THE LIST OF NEXT STARTING POINT *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *  *   FILE DEFINITION      *  *  *  *  *  *  *  *
CMS FILEDEF FILE1 DISK NO-ASR CARD A;
CMS FILEDEF FILE2 DISK ASR-INV TOT-MRGE A;
CMS FILEDEF OUTFILE1 DISK ADJ-LIST EXCESS (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF OUTFILE2 DISK ADJ-LIST ALL-BIL (RECFM F LRECL 80 BLOCK 80;
**********************************************************************
DATA DONE;
  INFILE FILE1;
  INPUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16;
   IF _N_ EQ 1 THEN DO;
      MCCNUM = 1;
      STARTAT = _N_;
      FILE OUTFILE1;
      PUT MCCNUM 1-3 MCC $5-7 STARTAT 9-12;
   END;
   ELSE IF MCC NE LAG(MCC) THEN DO;
      MCCNUM + 1;
      STARTAT = _N_;
      FILE OUTFILE1;
      PUT MCCNUM 1-3 MCC $5-7 STARTAT 9-12;
   END;
DATA DTWO;
  INFILE FILE2;
  INPUT MCC $1-3 MOS 5-8 GD 10 CNT 12-13 NUM 15-16;
   IF _N_ EQ 1 THEN DO;
      MCCNUM = 1;
      STARTAT = _N_;
      FILE OUTFILE2;
      PUT MCCNUM 1-3 MCC $5-7 STARTAT 9-12;
   END;
   ELSE IF MCC NE LAG(MCC) THEN DO;
      MCCNUM + 1;
      STARTAT = _N_;
      FILE OUTFILE2;
      PUT MCCNUM 1-3 MCC $5-7 STARTAT 9-12;
   END;
```

APPENDIX K
EXCESS FORTRAN

1. PROGRAM TO REMOVE EXCESS AND NON-MOVERS FROM ASR TO MAKE FREE ASR

```
***********************************************************************
*                                                                     *
*      *  *  *  *   PROGRAM NAME: EXCESS FORTRAN       *  *  *  *     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                 * * *   OVERVIEW AND PURPOSE    * * *               *
*                                                                     *
*    THIS PROGRAM SUBTRACTS THE NON-MOVERS FROM THE ASR WHENEVER THEY *
* CAN BE PLACED WITH CERTAINTY IN A PARTICULAR BILLET.  IT BEGINS BY  *
* REMOVING THE OBVIOUS MATCHES, AND PROCEEDS TO EMPLOY THE HEURISTICS *
* MENTIONED IN APPENDIX I (INVNTRY1 SAS) TO DETERMINE LESS APPARENT   *
* ONES.  EACH OF THE HEURISTICS IS EXPLAINED IN THE PROGRAM AS IT IS  *
* USED, HENCE THE PROGRAM IS SELF-DOCUMENTING.                        *
*    THE ONLY ONE OF THE OUTPUT FILES WHICH IS ACTUALLY USED IN THE   *
* PROTOTYPE IS THE FREE ASR WHICH CONTAINS THE DEMAND TO WHICH THE    *
* MOVERS WILL BE MATCHED.  THE OTHER TWO OUTPUT FILES ARE             *
* INFORMATIONAL.  NO-XCESS SUP-DEM CONTAINS A LIST OF DEMANDS WHICH   *
* ARE COMPLETELY FILLED BY NON-MOVERS.  EXCESS PERSONL CONTAINS A LIST *
* OF THOSE PERSONNEL WHO COULD NOT BE MATCHED TO AN ASR DEMAND AND WHO *
* ARE THEREFORE EXCESS.  IN FACT, SINCE THE E-CARDS WERE NOT USED IN  *
* THE PROTOTYPE TO PERFORM THE DETERMINATION OF WHO IS EXCESS, THIS   *
* FILE IS OF LIMITED USE.  HOWEVER, IF THE E-CARDS ARE USED IN THIS   *
* PROGRAM AT A LATER DATE, THE FILE COULD BE USED IN THE EXCESS       *
* RECONCILIATION PROCESS TO IDENTIFY NOT ONLY THOSE WHO ARE IMPROPERLY *
* ASSIGNED, BUT TO HELP IDENTIFY WHICH E-CARDS MIGHT NOT BE REFLECTING *
* THE DESIRES OF THE MONITORS.                                        *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                  ***   FILE DEFINITIONS    ***                      *
*                                                                     *
*   FILEDEF    FILE IDENTIFICATION        DESCRIPTION / PURPOSE       *
*     14         ADJ-LIST EXCESS      ADJACENCY LIST FOR FILEDEF 16   *
*     15         ADJ-LIST ALL-BIL     ADJACENCY LIST FOR FILEDEF 17   *
*     16         NO-ASR CARD          LIST OF NON-MOVERS IN BILLETS   *
*                                     FOR WHICH NO ASR MATCH IS FOUND *
*     17         ASR-INV TOT-MRGE     LIST OF MERGED ASR/NON-MOVERS   *
*     18         FREE ASR             DEMAND FOR MOVERS               *
*     19         NO-XCESS SUP-DEM     LIST OF ALL BILLETS ON ASR WHERE *
*                                     ENTIRE DEMAND IS MET BY NON-MOVRS*
*     20         EXCESS PERSONL       LIST OF PEOPLE WHO COULD NOT BE *
*                                     MATCHED TO SOME ASR DEMAND      *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                 ***   DEFINITION OF TERMS    ***                    *
*                                                                     *
*  BEGINA    START POINTER FOR PRESENT MCC GROUP WITHIN FILEDEF 14    *
*  BEGINB    START POINTER FOR THE NEXT MCC GROL. WITHIN FILEDEF 14   *
*  BGRD      BILLET GRADE (THE GRADE LISTED ON THE ASR FOR THE BILLET)*
*  BMOS      BILLET MOS (THE MOS LISTED ON THE ASR FOR THE BILLET)    *
*  CAPCTY    ASR DEMAND FOR THE MOS/GRADE                             *
*  EXCESS    THE NUMBER OF A PARTICULAR MOS/GRD FOUND IN FILEDEF 20   *
*  FLAGX     END OF FILE BINARY INDICATOR FOR FILEDEF 14              *
*  FLAGY     END OF FILE BINARY INDICATOR FOR FILEDEF 15              *
*  GOTILX    INDICATES THE LENGTH OF THE SEARCH THROUGH FILEDEF 14    *
*  GOTILY    INDICATES THE LENGTH OF THE SEARCH THROUGH FILEDEF 15    *
*   GRD        GRADE OR RANK                                                  *
*   HGRD       HIGHEST ACCEPTABLE GRADE FOR A DEMAND                          *
*   LGRD       LOWEST ACCEPTABLE GRADE FOR A DEMAND                           *
*   MARKER     COUNTER TO MARK POSITION IN THE OVERALL LIST                   *
*   MCCX1      CURRENT MCC FOR FILEDEF 14                                     *
*   MCCX2      NEXT MCC FOR FILEDEF 14                                        *
*   MCCY1      CURRENT MCC FOR FILEDEF 15                                     *
*   MCCY2      NEXT MCC FOR FILEDEF 15                                        *
*   PEOPLE     THE NUMBER OF PEOPLE ACTUALLY ON HAND WITH THAT MOS/GRD        *
*   PMOS       PRIMARY MOS                                                    *
*   STARTA     START POINTER FOR MOS/GRADE COMBINATION IN FILEDEF 14          *
*   STARTB     START POINTER FOR NEXT MOS/GRADE COMB. IN FILEDEF 14           *
*                                                                             *
*******************************************************************************
      INTEGER*4 FLAGX,FLAGY,STARTA,STARTB,BEGINA,BEGINB,GOTILX,GOTILY,
     CBMOS,BGRD,PEOPLE,CAPCTY,PMOS,GRD,EXCESS,FIRST2,MARKER,LGRD,HGRD
      CHARACTER*4 MCCX1, MCCX2,MCCY1,MCCY2
      DIMENSION BMOS(500),BGRD(500),PEOPLE(500),CAPCTY(500),PMOS(500),
     CGRD(500),EXCESS(500),FIRST2(500),MARKER(500)
***********          INITIALIZE          ********
      FLAGX = 0
      FLAGY = 0
***   READ FIRST VALUES TO BE COMPARED     ***
      READ(14,101) MCCX2,STARTB
      READ(15,101) MCCY2,BEGINB
101   FORMAT(4X,A3,1X,I4)
**  SET STARTING AND ENDING POINTS FOR CARDS TO BE SEARCHED IN EACH LIST
**  SET START AND END VALUES OF I (K) FOR SEARCH OF EXCESS FILE
1     MCCX1 = MCCX2
      STARTA = STARTB
      READ(14,101,END=91) MCCX2,STARTB
2     IF (FLAGX .EQ. 0) GOTILX = STARTB - STARTA
      IF (FLAGX .EQ. 1) GOTILX = STARTA + 200
**  SET STARTING AND ENDING POINTS FOR SEARCH OF ALL CARDS SORTED BY MCC
3     MCCY1 = MCCY2
      BEGINA = BEGINB
      READ(15,101,END=92) MCCY2,BEGINB
4     IF (FLAGY .EQ. 0) GOTILY = BEGINB - BEGINA
      IF (FLAGY .EQ. 1) GOTILY = BEGINA + 200
**  IF CARD CHOSEN NOT IN SAME MCC AS EXCESS CARD, GO THROUGH ALL CARDS
**  OF THAT MCC AND RECORD EACH MCC MOS GRADE AND DEMAND FOR THAT CARD,
**  SINCE THE CARD IS CORRECT.
      IF (MCCY1 .NE. MCCX1) THEN
         DO 10 I = 1,GOTILY
            READ(17,102,END=94)BMOS(I), BGRD(I),CAPCTY(I)
102         FORMAT(4X,I4,1X,I1,4X,I2)
            WRITE(18,103) MCCY1, BMOS(I),BGRD(I),CAPCTY(I)
103         FORMAT(A3,1X,I4,1X,I1,1X,I2)
10          CONTINUE
93          GO TO 3
**  IF THE MCC FROM THE COMPLETE LIST IS THE SAME AS THE MCC OF THE
**  EXCESS CARD THEN ALL OF THE FOLLOWING n ="GOTILY" CARDS FROM THE
**  COMPLETE LIST WILL ALSO BE FROM THE SAME MCC, AND MIGHT BE
**  USEFUL IN ELIMINATING ALL OR PART OF THE EXCESS AMOUNT, IF THEY
**  MATCH THE EXCESS CARD WITHIN ONE GRADE, OR FILL SOME OTHER
**  CRITERIA WHICH MAKES THE EXCESS AN ACCEPTABLE FILL FOR THAT
**  BILLET.
      ELSE IF (MCCX1 .EQ. MCCY1) THEN
**  READ ALL THE CARDS FROM THAT MCC WHICH APPEAR ON THE COMPLETE LIST.
**  MARKER IS USED TO ENSURE THAT RECORDS WHICH APPEAR ON BOTH
**  FILES DO NOT APPEAR TWICE IN THE OUTPUT.
         DO 20 I = 1,GOTILY
            READ(17,104,END=94)BMOS(I), BGRD(I),PEOPLE(I),CAPCTY(I)
104         FORMAT(4X,I4,1X,I1,1X,I2,1X,I2)
            MARKER(I) = 0
20          CONTINUE
```

```
**  READ ALL THE EXCESS CARDS FROM THAT MCC
5         DO 30 J = 1,GOTILX
               READ(16,105,END=95) PMOS(J),GRD(J),EXCESS(J),FIRST2(J)
105         FORMAT(4X,I4,1X,I1,1X,I2,4X,I2)
30          CONTINUE
**  BEGINNING WITH THE FIRST EXCESS CARD FROM THAT MCC, SEE IF ANY OF
**  THE OTHER CARDS AT THAT MCC COULD BE USED TO "RECEIVE"
**  OR ABSORB SOME OF THAT EXCESS.
6       DO 40 K = 1,GOTILX
             DO 50 L = 1,GOTILY
**  FIRST, CHECK TO SEE IF THE CARD WE ARE COMPARING IN THE TOTAL
**  FILE IS THE SAME AS THE EXCESS FILE WE ARE LOOKING AT.
               IF ((BMOS(L) .EQ. PMOS(K)) .AND. (BGRD(L) .EQ.
    C       GRD(K))) MARKER(L) = 1
**  IF THE CAPACITY ON THE CARD IS ZERO, IT OBVIOUSLY CANNOT ABSORB
**  ANY OF THE EXCESS.  WHEN WE SEND THE PROGRAM TO 7 (GOTO 7), WE
**  ARE SELECTING THE NEXT CARD FROM THE TOTAL CARD FILE.
               IF (CAPCTY(L) .EQ. 0) GOTO 7
**  IF THE DEMAND IS MORE THAN ONE GRADE AWAY FROM THE DEMAND ON THE
**  EXCESS CARD, IT CANNOT MEET THE GRADE SUBSTITUTION CRITERIA
               LGRD = BGRD(L) - 1
               HGRD = BGRD(L) + 1
               IF ((GRD(K) .LT. LGRD) .OR. (GRD(K) .GT. HGRD)) GO TO 7
**  3060 AND 3070 MOS'S ARE INTERCHANGEABLE ON THE E1 CARDS, SO ANY
**  3060 OR 3070 EXCESSES MAY BE ABSORBED BY 3070 OR 3060 BILLETS
               IF (((PMOS(K) .EQ. 3060) .OR. (PMOS(K) .EQ. 3070)).AND.
    C       ((BMOS(L) .EQ. 3060) .OR. (BMOS(L) .EQ.3070)))  THEN
                  IF (CAPCTY(L) .GE. EXCESS(K)) THEN
                     CAPCTY(L) = CAPCTY(L) - EXCESS(K)
                     EXCESS(K) = 0
                     WRITE(19,103) MCCX1, PMOS(K),GRD(K),EXCESS(K)
                     GOTO 40
                  ELSE IF (CAPCTY(L) .LT. EXCESS(K)) THEN
                     EXCESS(K) = EXCESS(K) - CAPCTY(L)
                     CAPCTY(L) = 0
                     GOTO 7
                  END IF
               END IF
**  SINCE THE SECONDARY MOS'S WERE NOT PULLED FROM THE INVENTORY
**  NONE OF THE 7596 BMOS'S WERE MATCHED BY THE SAS JOB.
**  IT IS ASSUMED THAT IF THERE IS ARE 75XX MOS'S AT AN MCC, THAT
**  THERE IS AT LEAST ONE WITH THE 7596 SECONDARY FOR EACH 7596 BILLET.
               IF ((FIRST2(K) .EQ. 75) .AND. (BMOS(L) .EQ. 7596)) THEN
                  IF (CAPCTY(L) .GE. EXCESS(K)) THEN
                     CAPCTY(L) = CAPCTY(L) - EXCESS(K)
                     EXCESS(K) = 0
                     WRITE(19,103) MCCX1, PMOS(K),GRD(K),EXCESS(K)
                     GOTO 40
                  ELSE IF (CAPCTY(L) .LT. EXCESS(K)) THEN
                     EXCESS(K) = EXCESS(K) - CAPCTY(L)
                     CAPCTY(L) = 0
                     GOTO 7
                  END IF
               END IF
**  SINCE 9912 IS ONLY A BMOS, ALL 9912 BMOS ASR DEMANDS WERE LEFT
**  UNMET.  HERE, WE ALLOW ANY REMAINING 75XX'S TO FILL 9912 BILLETS
**  NOTE - WE SHOULD LATER MARK ALL SHORT 75XX MOS'S AND ENSURE THAT
**  SHORT MOS'S ARE NOT ALLOWED TO FILL 9912 BILLETS.  THIS CAN BE DONE
**  BY MARKING THE SHORT MOS'S SOMEHOW.
               IF ((FIRST2(K) .EQ. 75) .AND. (BMOS(L) .EQ. 9912)) THEN
                  IF (CAPCTY(L) .GE. EXCESS(K)) THEN
                     CAPCTY(L) = CAPCTY(L) - EXCESS(K)
                     EXCESS(K) = 0
                     WRITE(19,103) MCCX1, PMOS(K),GRD(K),EXCESS(K)
                     GOTO 40
                  ELSE IF (CAPCTY(L) .LT. EXCESS(K)) THEN
                     EXCESS(K) = EXCESS(K) - CAPCTY(L)
```

```
                    CAPCTY(L) = 0
                    GOTO 7
                 END IF
              END IF
              IF ((FIRST2(K) .EQ. 75) .AND. (BMOS(L) .EQ. 9958)) THEN
                    CAPCTY(L) = CAPCTY(L) - 1
                    EXCESS(K) = EXCESS(K) - 1
                    IF (EXCESS(K) .EQ. 0) THEN
                       WRITE(19,103) MCCX1, PMOS(K),GRD(K),EXCESS(K)
                       GOTO 40
                    END IF
                    IF (CAPCTY(L) .EQ. 0) GOTO 7
              END IF
7     CONTINUE
50    CONTINUE
**  PRINT ANY EXCESSES WHICH DID NOT GET ABSORBED BY THE OTHER BILLETS
**  AT THE MCC
       WRITE(20,107) MCCX1,PMOS(K),GRD(K),EXCESS(K)
40    CONTINUE
**  NOW ALL THE ASR DEMAND AT THE MCC HAS BEEN SEARCHED IN AN
**  EFFORT TO REDUCE SOME OF THE EXCESSES.  THOSE EXCESSES WHICH COULD
**  BE ELIMINATED WERE ALREADY OUTPUT TO THE FILE CONTAINING ALL OF THE
**  NON-EXCESS DEMAND.  ALL REMAINING EXCESSES WERE OUTPUT TO A FILE
**  OF EXCESSES. (FILEDEF 10 - EXCESS PERSONL A1).  NOW, WE WILL OUTPUT
**  THE REST OF THE CARDS FROM THE TOTAL ASR DEMAND FILE EITHER TO A
**  NULL FILE FOR THOSE WITH NO REMAINING CAPACITY (FILEDEF 09 -
**  NO-XCESS SUP-DEM A1) OR TO A FILE CONTAINING THE TRUE ASR DEMAND
**  REMAINING FOR MOVERS. (FILEDEF 08 - FREE ASR A1)   IN DOING SO, WE
**  WILL OF COURSE, LEAVE OFF THOSE BILLETS WHICH WE ALREADY OUTPUTED
**  TO ONE OF THE OTHER FILES, WHICH ALSO CORRESPOND TO THE BILLETS ON
**  THE EXCESS FILE (AND HENCE ARE MARKED BY MARKER(L).
       DO 60 N = 1,GOTILY
       IF (CAPCTY(N) .EQ. 0) THEN
          IF (MARKER(N) .EQ. 0) THEN
            WRITE(19,107)MCCY1,BMOS(N), BGRD(N),CAPCTY(N)
          END IF
       ELSE IF (CAPCTY(N) .NE. 0) THEN
          IF (MARKER(N) .EQ. 0) THEN
            WRITE(18,107)MCCY1,BMOS(N), BGRD(N),CAPCTY(N)
          END IF
       END IF
107        FORMAT(A3,1X,I4,1X,I1,1X,I2)
60    CONTINUE
       END IF
       IF (FLAGX .NE.1) GOTO 1
       GO TO 9
91    FLAGX = 1
       GO TO 2
92    FLAGY = 1
       GO TO 4
94    GOTILY = I
       GOTO 5
95    GOTILX = J
       GO TO 6
9     CLOSE(14)
       CLOSE(15)
       CLOSE(16)
       CLOSE(17)
       CLOSE(18)
       CLOSE(19)
       CLOSE(20)
       STOP
       END
```

APPENDIX L
ASRE2A SAS

1. PROGRAM TO ATTACH BILLET OFFICER DESCRIPTIONS (BOD'S) TO FREE ASR

```
**********************************************************************
*                                                                    *
*       * * * *   PROGRAM NAME: ASRE2A SAS         * * * *           *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*               * * *   OVERVIEW AND PURPOSE   * * *                 *
*                                                                    *
*    THIS PROGRAM ATTACHES THE BILLET OFFICER DESCRIPTION (BOD) TO   *
* EACH ASR DEMAND THAT APPEARS IN THE FREE ASR.   IT DOES THIS BY    *
* MATCHING EACH OF THE E2 CARDS THAT MATCH TO A SPECIFIC DEMAND TO THE *
* DEMAND IN ORDER OF THEIR PRIORITY.  THE EXTREME LENGTH OF THIS     *
* PROGRAM IS A RESULT OF ONE OF THE MAJOR DEFICIENCIES WITH USING SAS *
* TO MATCH AND MERGE SEVERAL FILES.    SAS CANNOT USE THE LAG FUNCTION *
* ON VARIABLES CREATED WITHIN THE DATA SET.  THEREFORE WE MUST       *
* MAKE AN ADDITIONAL DATA SET WHICH WILL BE USED TO PERFORM THE DEMAND *
* RECONCILIATION PROCESS.  ALSO, IT CANNOT INPUT DATA FROM MULTIPLE  *
* SETS SIMULTANEOUSLY.  RATHER, IT MUST COMPLETELY READ EACH SET IN  *
* BEFORE READING THE FIRST VALUE IN THE NEXT SET.   AS A RESULT, THIS *
* PROGRAM HAS NUMEROUS LOOPS THROUGH THE SAME DATA SET IN ORDER TO   *
* CHECK POSSIBLE ADDITIONAL REQUIREMENTS AT A PARTICULAR MCC.  IN ANY *
* FULL SCALE MODEL, IT WOULD BE ADVISEABLE TO RE-WRITE THIS ROUTINE IN *
* FORTRAN.                                                           *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                    *
*               ***   DEFINITION OF TERMS   ***                      *
*                                                                    *
*   A         FIST CHARACTER OF THE MCC                              *
*   B         SECOND CHARACTER OF THE MCC                            *
*   BOD       BILLET OFFICER DESCRIPTION                             *
*   C         THIRD CHARACTER OF THE MCC                             *
*   CAP       TOTAL AUTHORIZED DEMAND AT THE CURRENT MCC             *
*   CAP2      TOTAL AUTHORIZED DEMAND AT THE NEXT MCC                *
*   D         FIRST TWO CHARACTERS OF THE MCC                        *
*   GD        GRADE OR RANK (INTEGER)                                *
*   GRD       GRADE OR RANK (CHARACTER)                              *
*   GRD2      GRADE OR RANK (INTEGER) OF NEXT DEMAND                 *
*   MOS       THE MOS CURRENTLY BEING LOOKED AT                      *
*   MOS2      THE NEXT MOS WHICH WILL BE LOOKED AT                   *
*   MCC       CURRENT MONITORED COMMAND CODE (LOCATION)              *
*   MCC2      NEXT MONITORED COMMAND CODE (LOCATION)                 *
*   NUM       NUMBER OF MARINES FILLING THE DEMAND BEING LOOKED AT   *
*   NUM2      NUMBER OF MARINES FILLING THE NEXT DEMAND TO BE EXAMINED *
*   OT        OFFICER TYPE                                           *
*   SA        SPLIT ADJUSTMENT - ADD (A), SUBTRACT (S), OR PROPORT (P) *
*   SPL       STAFFING PRECEDENCE LEVEL                              *
*   SAMT      SPLIT AMOUNT   NAME CONTAINING START & ENDING POINTS   *
*                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *  *  FILE DEFINITION  *  *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK WKE2 CRD A;
CMS FILEDEF FTWO DISK FREE ASR A;
CMS FILEDEF FTHREE DISK E2ASR MIX-MOVR (RECFM F LRECL 80 BLOCK 80;
**********************************************************************
*READ IN E2 CARDS;
DATA DONE
      DTWO
```

```
        DTHREE
        DFOUR ;
     INFILE FONE;
     INPUT MOS 1-4 OT $ 7-14 A $ 17 D $ 17-18 B $ 18 C $ 19 MCC $ 17-19
              GRD $21-22 SA $ 24 SAMT 25-27 SPL 29 BOD $ 34-42;
     E2NUM + 1;
     IF GRD = 'WO' THEN GD = 1;
     IF GRD = '02' THEN GD = 2;
     IF GRD = '03' THEN GD = 3;
     IF GRD = '04' THEN GD = 4;
     IF GRD = '05' THEN GD = 5;
*BREAK E2 CARDS INTO FOUR GROUPS;
     IF A = '*' THEN OUTPUT DONE;
     ELSE IF B = '*' THEN OUTPUT DTWO;
     ELSE IF C = '*' THEN OUTPUT DTHREE;
     ELSE OUTPUT DFOUR;
*INPUT DATA FROM ASR;
DATA DFIVE (KEEP = MOS GD MCC A D NUM);
     INFILE FTWO;
     INPUT A $1 D $1-2 MCC $1-3 MOS 5-8 GD 10 NUM 12-13;
*MERGE ASR WITH ALL OF E2 CARDS;
*
PROC SORT DATA = DFIVE;
     BY MOS GD MCC;
*SORT FULL SPEC MCC E2 CARDS;
PROC SORT DATA = DFOUR;
     BY MOS GD MCC;
*MERGE FULL SPEC MCC E2 CARDS WITH FREE ASR;
DATA DSIX;
     MERGE DFOUR DFIVE;
     BY MOS GD MCC;
*SORT 3STAR MCC E2 CARDS;
PROC SORT DATA = DONE;
     BY MOS GD;
*MERGE 3STAR MCC E2 CARDS WITH FREE ASR;
DATA DSEVEN;
     MERGE DONE DFIVE;
     BY MOS GD;
*SORT 2STAR MCC E2 CARDS;
PROC SORT DATA = DTWO;
     BY MOS GD A;
*MERGE 2STAR MCC E2 CARDS WITH FREE ASR;
DATA DEIGHT;
     MERGE DTWO DFIVE;
     BY MOS GD A;
*SORT 1STAR MCC E2 CARDS;
PROC SORT DATA = DTHREE;
     BY MOS GD D;
*MERGE 1STAR MCC E2 CARDS WITH FREE ASR;
DATA DNINE ;
     MERGE DTHREE DFIVE;
     BY MOS GD D;
*OUTPUT DATA;
DATA DTEN;
     SET DSIX DSEVEN DEIGHT DNINE;
     IF NUM EQ '.' THEN DELETE;
     IF BOD EQ '          ' THEN DELETE;
     CAP = NUM;
PROC SORT DATA = DTEN;
       BY MOS GD MCC SPL;
DATA D11;
     SET DTEN;
       IF MOS NE LAG(MOS) OR GD NE LAG(GD) OR MCC NE LAG(MCC) THEN DO;
             IF SA NE 'A' THEN CAP = NUM;
```

```
          ELSE IF SA EQ 'A' THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
        END;
    DATA D12;
      SET D11;
          NUM2 = LAG(NUM) - LAG(CAP);
          MOS2 = LAG(MOS);
          GRD2 = LAG(GD);
          MCC2 = LAG(MCC);
          IF SA NE 'A' THEN DO;
            IF _N_ = 1 THEN CAP = NUM;
            IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              CAP = NUM;
            END;
            IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
            CAP = NUM;
           END;
          END;
          ELSE IF SA EQ 'A' THEN DO;
            IF _N_ = 1 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
          END;
    DATA D13;
      SET D12;
          NUM2 = LAG(NUM) - LAG(CAP);
          MOS2 = LAG(MOS);
          GRD2 = LAG(GD);
          MCC2 = LAG(MCC);
          IF SA NE 'A' THEN DO;
            IF _N_ = 1 THEN CAP = NUM;
            IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              CAP = NUM;
            END;
            IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
            CAP = NUM;
           END;
          END;
          ELSE IF SA EQ 'A' THEN DO;
            IF _N_ = 1 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
          END;
    DATA D15;
```

```
   SET D13;
        NUM2 = LAG(NUM) - LAG(CAP);
        MOS2 = LAG(MOS);
        GRD2 = LAG(GD);
        MCC2 = LAG(MCC);
        IF SA NE 'A' THEN DO;
          IF _N_ = 1 THEN CAP = NUM;
          IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
            NUM = NUM2;
            CAP = NUM;
          END;
          IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
          CAP = NUM;
         END;
        END;
        ELSE IF SA EQ 'A' THEN DO;
          IF _N_ = 1 THEN DO;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
          ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
            NUM = NUM2;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
          ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
        END;
DATA D14;
   SET D15;
        NUM2 = LAG(NUM) - LAG(CAP);
        MOS2 = LAG(MOS);
        GRD2 = LAG(GD);
        MCC2 = LAG(MCC);
        IF SA NE 'A' THEN DO;
          IF _N_ = 1 THEN CAP = NUM;
          IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
            NUM = NUM2;
            CAP = NUM;
          END;
          IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
          CAP = NUM;
         END;
        END;
        ELSE IF SA EQ 'A' THEN DO;
          IF _N_ = 1 THEN DO;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
          ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
            NUM = NUM2;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
          ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
            IF SAMT GE NUM THEN CAP = NUM;
            IF SAMT LT NUM THEN CAP = SAMT;
          END;
        END;
DATA DELEVEN;
   SET D14;
        NUM2 = LAG(NUM) - LAG(CAP);
        MOS2 = LAG(MOS);
        GRD2 = LAG(GD);
        MCC2 = LAG(MCC);
        IF SA NE 'A' THEN DO;
          IF _N_ = 1 THEN CAP = NUM;
```

```
            IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              CAP = NUM;
            END;
            IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
            CAP = NUM;
           END;
          END;
          ELSE IF SA EQ 'A' THEN DO;
            IF _N_ = 1 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
            ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
            END;
          END;
          IF CAP = 0 THEN DELETE;
          OLDCAP = LAG(CAP);
    PROC SORT DATA = DELEVEN;
       BY E2NUM;
  DATA _NULL_;
   SET DELEVEN;
   FILE FTHREE;
   PUT MOS 1-4 OT 7-14 MCC 17-19 GD 21-22
       SA 24 SAMT 26-28 SPL 30 BOD 34-42 CAP 44-46 NUM 48-50;
```

APPENDIX M
ASRE2B SAS

1. PROGRAM TO ATTACH BILLET OFFICER DESCRIPTIONS (BOD'S) TO FIXED ASR

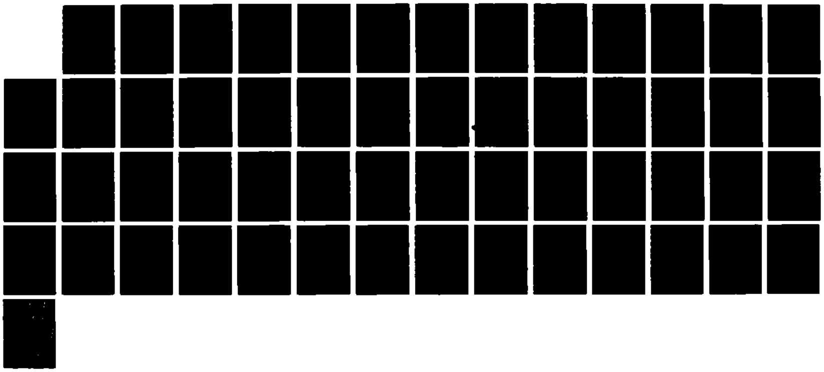
```
*********************************************************************
*                                                                   *
*       *  *  *  *   PROGRAM NAME: ASRE2B SAS         *  *  *  *    *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                   *
*                * * *   OVERVIEW AND PURPOSE    * * *              *
*                                                                   *
*    THIS PROGRAM ATTACHES THE BILLET OFFICER DESCRIPTION (BOD) TO  *
* EACH ASR DEMAND THAT APPEARS IN THE FIXED (WORK) ASR. THIS IS DONE *
* THE SAME WAY THAT BODS ARE ATTACHED TO THE FREE DEMAND IN ASRE2A: BY *
* MATCHING EACH OF THE E2 CARDS THAT MATCH TO A SPECIFIC DEMAND TO THE *
* DEMAND IN ORDER OF THEIR PRIORITY.  THE EXTREME LENGTH OF THIS    *
* PROGRAM IS A RESULT OF ONE OF THE MAJOR DEFICIENCIES WITH USING SAS *
* TO MATCH AND MERGE SEVERAL FILES.    SAS CANNOT USE THE LAG FUNCTION *
* ON VARIABLES CREATED WITHIN THE DATA SET.  THEREFORE WE MUST      *
* MAKE AN ADDITIONAL DATA SET WHICH WILL BE USED TO PERFORM THE DEMAND *
* RECONCILIATION PROCESS.  ALSO, IT CANNOT INPUT DATA FROM MULTIPLE *
* SETS SIMULTANEOUSLY.  RATHER, IT MUST COMPLETELY READ EACH SET IN *
* BEFORE READING THE FIRST VALUE IN THE NEXT SET.   AS A RESULT, THIS *
* PROGRAM HAS NUMEROUS LOOPS THROUGH THE SAME DATA SET IN ORDER TO  *
* CHECK POSSIBLE ADDITIONAL REQUIREMENTS AT A PARTICULAR MCC.  IN ANY *
* FULL SCALE MODEL, IT WOULD BE ADVISEABLE TO RE-WRITE THIS ROUTINE IN *
* FORTRAN.  THE ONLY DIFFERENCES BETWEEN THIS PROGRAM AND ASRE2A ARE *
* THE FILE DEFINITIONS.                                             *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                   *
*                ***   DEFINITION OF TERMS   ***                    *
*                                                                   *
*   A         FIST CHARACTER OF THE MCC                             *
*   B         SECOND CHARACTER OF THE MCC                           *
*   BOD       BILLET OFFICER DESCRIPTION                            *
*   C         THIRD CHARACTER OF THE MCC                            *
*   CAP       TOTAL AUTHORIZED DEMAND AT THE CURRENT MCC            *
*   CAP2      TOTAL AUTHORIZED DEMAND AT THE NEXT MCC               *
*   D         FIRST TWO CHARACTERS OF THE MCC                       *
*   GD        GRADE OR RANK (INTEGER)                               *
*   GRD       GRADE OR RANK (CHARACTER)                             *
*   GRD2      GRADE OR RANK (INTEGER) OF NEXT DEMAND                *
*   MOS       THE MOS CURRENTLY BEING LOOKED AT                     *
*   MOS2      THE NEXT MOS WHICH WILL BE LOOKED AT                  *
*   MCC       CURRENT MONITORED COMMAND CODE (LOCATION)             *
*   MCC2      NEXT MONITORED COMMAND CODE (LOCATION)                *
*   NUM       NUMBER OF MARINES FILLING THE DEMAND BEING LOOKED AT  *
*   NUM2      NUMBER OF MARINES FILLING THE NEXT DEMAND TO BE EXAMINED *
*   OT        OFFICER TYPE                                          *
*   SA        SPLIT ADJUSTMENT - ADD (A), SUBTRACT (S), OR PROPORT (P) *
*   SPL       STAFFING PRECEDENCE LEVEL                             *
*   SAMT      SPLIT AMOUNT   NAME CONTAINING START & ENDING POINTS  *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *  *  *  FILE DEFINITION  *  *  *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK WKE2 CRD A;
CMS FILEDEF FTWO DISK WORK ASR A;
CMS FILEDEF FTHREE DISK E2ASR MIX-NMOV (RECFM F LRECL 80 BLOCK 80;
*READ IN E2 CARDS;
DATA DONE
```

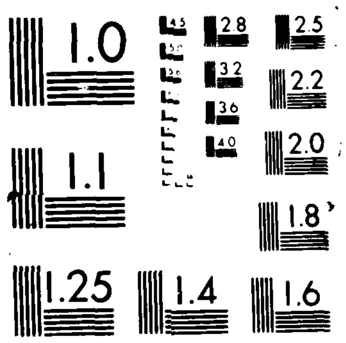
```
        DTWO
        DTHREE
        DFOUR ;
    INFILE FONE;
    INPUT MOS 1-4 OT $ 7-14 A $ 17 D $ 17-18 B $ 18 C $ 19 MCC $ 17-19
          GRD $21-22 SA $ 24 SAMT 25-27 SPL 29 BOD $ 34-42;
    E2NUM + 1;
    IF GRD = 'WO' THEN GD = 1;
    IF GRD = '02' THEN GD = 2;
    IF GRD = '03' THEN GD = 3;
    IF GRD = '04' THEN GD = 4;
    IF GRD = '05' THEN GD = 5;
  *BREAK E2 CARDS INTO FOUR GROUPS;
    IF A = '*' THEN OUTPUT DONE;
    ELSE IF B = '*' THEN OUTPUT DTWO;
    ELSE IF C = '*' THEN OUTPUT DTHREE;
    ELSE OUTPUT DFOUR;

  *INPUT DATA FROM ASR;
  DATA DFIVE (KEEP = MOS GD MCC A D NUM);
    INFILE FTWO;
    INPUT MOS 1-4 A $7 D $7-8 MCC $7-9 GD 12 NUM 15-19;
  *MERGE ASR WITH ALL OF E2 CARDS;
  *    ;
  PROC SORT DATA = DFIVE;
    BY MOS GD MCC;
  *SORT FULL SPEC MCC E2 CARDS;
  PROC SORT DATA = DFOUR;
    BY MOS GD MCC;
  *MERGE FULL SPEC MCC E2 CARDS WITH FREE ASR;
  DATA DSIX;
    MERGE DFOUR DFIVE;
    BY MOS GD MCC;
  *SORT 3STAR MCC E2 CARDS;
  PROC SORT DATA = DONE;
    BY MOS GD;
  *MERGE 3STAR MCC E2 CARDS WITH FREE ASR;
  DATA DSEVEN;
    MERGE DONE DFIVE;
    BY MOS GD;
  *SORT 2STAR MCC E2 CARDS;
  PROC SORT DATA = DTWO;
    BY MOS GD A;
  *MERGE 2STAR MCC E2 CARDS WITH FREE ASR;
  DATA DEIGHT;
    MERGE DTWO DFIVE;
    BY MOS GD A;
  *SORT 1STAR MCC E2 CARDS;
  PROC SORT DATA = DTHREE;
    BY MOS GD D;
  *MERGE 1STAR MCC E2 CARDS WITH FREE ASR;
  DATA DNINE ;
    MERGE DTHREE DFIVE;
    BY MOS GD D;
  *OUTPUT DATA;
  DATA DTEN;
    SET DSIX DSEVEN DEIGHT DNINE;
    IF NUM EQ '.' THEN DELETE;
    IF BOD EQ '         ' THEN DELETE;
    CAP = NUM;
  PROC SORT DATA = DTEN;
     BY MOS GD MCC SPL;
  DATA D11;
    SET DTEN;
```

```
      IF MOS NE LAG(MOS) OR GD NE LAG(GD) OR MCC NE LAG(MCC) THEN DO;
         IF SA NE 'A' THEN CAP = NUM;
         ELSE IF SA EQ 'A' THEN DO;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
       END;
   DATA D12;
     SET D11;
         NUM2 = LAG(NUM) - LAG(CAP);
         MOS2 = LAG(MOS);
         GRD2 = LAG(GD);
         MCC2 = LAG(MCC);
         IF SA NE 'A' THEN DO;
           IF _N_ = 1 THEN CAP = NUM;
           IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
             NUM = NUM2;
             CAP = NUM;
           END;
           IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
           CAP = NUM;
          END;
         END;
         ELSE IF SA EQ 'A' THEN DO;
           IF _N_ = 1 THEN DO;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
             NUM = NUM2;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
         END;
   DATA D13;
     SET D12;
         NUM2 = LAG(NUM) - LAG(CAP);
         MOS2 = LAG(MOS);
         GRD2 = LAG(GD);
         MCC2 = LAG(MCC);
         IF SA NE 'A' THEN DO;
           IF _N_ = 1 THEN CAP = NUM;
           IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
             NUM = NUM2;
             CAP = NUM;
           END;
           IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
           CAP = NUM;
          END;
         END;
         ELSE IF SA EQ 'A' THEN DO;
           IF _N_ = 1 THEN DO;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
             NUM = NUM2;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
             IF SAMT GE NUM THEN CAP = NUM;
             IF SAMT LT NUM THEN CAP = SAMT;
           END;
         END;
```

AD-A186 305 A PROTOTYPE DECISION SUPPORT SYSTEM FOR MARINE CORPS OFFICER ALLOCATION POLICY ANALYSIS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA P J EXNER SEP 87 2/2

UNCLASSIFIED F/G 5/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

```
DATA D15;
  SET D13;
       NUM2 = LAG(NUM) - LAG(CAP);
       MOS2 = LAG(MOS);
       GRD2 = LAG(GD);
       MCC2 = LAG(MCC);
       IF SA NE 'A' THEN DO;
         IF _N_ = 1 THEN CAP = NUM;
         IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
           NUM = NUM2;
           CAP = NUM;
         END;
         IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
         CAP = NUM;
        END;
       END;
       ELSE IF SA EQ 'A' THEN DO;
         IF _N_ = 1 THEN DO;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
         ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
           NUM = NUM2;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
         ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
       END;
DATA D14;
  SET D15;
       NUM2 = LAG(NUM) - LAG(CAP);
       MOS2 = LAG(MOS);
       GRD2 = LAG(GD);
       MCC2 = LAG(MCC);
       IF SA NE 'A' THEN DO;
         IF _N_ = 1 THEN CAP = NUM;
         IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
           NUM = NUM2;
           CAP = NUM;
         END;
         IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
         CAP = NUM;
        END;
       END;
       ELSE IF SA EQ 'A' THEN DO;
         IF _N_ = 1 THEN DO;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
         ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
           NUM = NUM2;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
         ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
           IF SAMT GE NUM THEN CAP = NUM;
           IF SAMT LT NUM THEN CAP = SAMT;
         END;
       END;
DATA DELEVEN;
  SET D14;
       NUM2 = LAG(NUM) - LAG(CAP);
       MOS2 = LAG(MOS);
       GRD2 = LAG(GD);
       MCC2 = LAG(MCC);
```

```
        IF SA NE 'A' THEN DO;
           IF _N_ = 1 THEN CAP = NUM;
           IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              CAP = NUM;
           END;
           IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
           CAP = NUM;
          END;
        END;
        ELSE IF SA EQ 'A' THEN DO;
           IF _N_ = 1 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS EQ MOS2 AND GD EQ GRD2 AND MCC EQ MCC2 THEN DO;
              NUM = NUM2;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
           END;
           ELSE IF MOS NE MOS2 OR GD NE GRD2 OR MCC NE MCC2 THEN DO;
              IF SAMT GE NUM THEN CAP = NUM;
              IF SAMT LT NUM THEN CAP = SAMT;
           END;
        END;
        IF CAP = 0 THEN DELETE;
        OLDCAP = LAG(CAP);
  PROC SORT DATA = DELEVEN;
      BY E2NUM;
DATA _NULL_;
 SET DELEVEN;
 FILE FTHREE;
 PUT MOS 1-4 OT 7-14 MCC 17-19 GD 21-22
     SA 24 SAMT 26-28 SPL 30 BOD 34-42 CAP 44-46 NUM 48-50;
```

APPENDIX N
E2ASRE1A SAS

1. PROGRAM TO ATTACH SUBSTITUTION LISTS TO FREE (MOVER) DEMANDS

```
***********************************************************************
*                                                                     *
*     * * * *   PROGRAM NAME: E2ASRE1A SAS        * * * *            *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                  * * *  OVERVIEW AND PURPOSE   * * *               *
*                                                                     *
*    THIS PROGRAM ATTACHES THE E1 CARD INFORMATION AND COST CODE      *
* INDICES TO THE FILE CONTAINING THE FREE ASR-E2 CARDS INFORMATION.   *
* THE FIRST TASK IS ATTACHING THE E1 CARD SUBSTITUTION LISTS TO THE   *
* DEMAND USING THE BOD'S WHICH WERE ATTACHED IN ASRE2A AS A MEANS OF  *
* LINKING THE DEMAND TO THE SUBSTITUTION LISTS.  THIS RESULTS IN A    *
* FILE WHICH HAS THE DEMAND BROKEN INTO SPECIFIC BILLETS (BY THE E2   *
* CARDS) WITH A SET OF UP TO FIVE SUBSTITUTION CRITERIA ATTACHED TO   *
* EACH SPECIFIC DEMAND (FROM THE E1CARDS.)                            *
*    THE NEXT TASK PERFORMED IN THIS PROGRAM IS THE PLACING OF THE    *
* APPROPRIATE COST CODE CENTER INDEX NEXT TO EACH DEMAND.  THIS       *
* CORRESPONDS, ON THE DEMAND SIDE, TO THE COST CODE CENTER INDEX WHICH *
* WAS ATTACHED TO EACH INDIVIDUAL IN FREE-FIX SAS, ON THE SUPPLY SIDE. *
*    MOST OF THE FUNCTIONS AND TASKS PERFORMED IN THE PROGRAM ARE     *
* EXPLAINED AS THEY ARE PERFORMED, THUS THE BODY OF THE PROGRAM IS    *
* SELF-DOCUMENTING.                                                   *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                  ***   DEFINITION OF TERMS   ***                    *
*                                                                     *
*   AMOS1     FIRST AMOS APPEARING ON THE E1 CARD (SUBSTITUTION)      *
*   AMOS2     SECOND AMOS APPEARING ON THE E1 CARD (SUBSTITUTION)     *
*   BOD       BILLET OFFICER DESCRIPTION                              *
*   BODNUMA   INDEX ON THE BOD FOUND IN THE E1 CARDS                  *
*   BODNUMB   INDEX ON THE BOD FOUND IN THE E2 CARDS                  *
*   CAPACITY  TOTAL AUTHORIZED DEMAND AT THE CURRENT MCC              *
*   COSTCTR   COST CENTER CODE INDEX                                  *
*   E1NUM     INDEX ON THE E1 CARDS                                   *
*   E2NUM     INDEX ON THE E2 CARDS                                   *
*   FITLVL    FIT LEVEL DEFINED BY ORDER OF SUBSTITUTION PREFERENCE   *
*   GD        GRADE OR RANK (INTEGER)                                 *
*   HEXP      EXPERIENCE CODE ASSOCIATED WITH HIGH GRADE              *
*   HGRD      HIGHEST ACCEPTABLE GRADE FOR A SUBSTITUTION             *
*   LDO       DUTY RESTRICTION CODE                                   *
*   LEXP      EXPERIENCE CODE ASSOCIATED WITH LOW GRADE               *
*   LGRD      LOWEST ACCEPTABLE GRADE FOR A SUBSTITUTION              *
*   MAC       MONITORED ACTIVITY CODE                                 *
*   MCC       CURRENT MONITORED COMMAND CODE (LOCATION)               *
*   MCC2      NEXT MONITORED COMMAND CODE (LOCATION)                  *
*   NUM       NUMBER OF MARINES FILLING THE DEMAND BEING LOOKED AT    *
*   OT        OFFICER TYPE                                            *
*   PMOS      PRIMARY MOS APPEARING ON AN E1 CARD (SUBSTITUTION)      *
*   SA        SPLIT ADJUSTMENT - ADD (A), SUBTRACT (S), OR PROPORT (P) *
*   SAMT      SPLIT AMOUNT   NAME CONTAINING START & ENDING POINTS    *
*   SEX       SEX CODE RESTRICTION INDICATOR                          *
*   SPL       STAFFING PRECEDENCE LEVEL                               *
*   T1 - T9   0/1 FLAGS FOR OFFICER CODE TYPES FOUND IN E1 CARDS      *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *    FILE DEFINITION   *  *  *  *  *  *  *  *  *  *
```

```
CMS FILEDEF DATAIN1 DISK WKE1 CRD A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN2 DISK E2ASR MIX-MOVR A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN3 DISK MOVR-EP ARRAY A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN4 DISK CCC-MCC CONVERT A1;
CMS FILEDEF DATAOUT1 DISK E1-INDEX CRD-A A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT2 DISK E2-INDEX CRD-A A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT3 DISK MOVR-DEM INPUT A (RECFM F LRECL 80 BLOCK 80;
*******************************************************************************
OPTIONS LINESIZE=80;
    /*
      THE FIRST PORTION OF THIS SAS JOB PUTS 2 INDICES ON THE E1 CARDS.
THE FIRST INDEX SIMPLY IS A COUNT OF THE E1 CARDS AND CAN BE USED TO
PUT THE CARDS BACK IN ORDER AFTER ANY COMBINATION OF OTHER SORTS ARE
PERFORMED ON THE FILE.  THE SECOND INDEX MARKS ALL THOSE E1 CARDS WHICH
HAVE THE SAME BOD.  THIS INDEX CAN BE USED TO RESTORE THE ORDER OF THE
E1 CARDS BY BOD AFTER THE FILE HAS BEEN SORTED SOME OTHER WAY.
      */;
DATA FILE1;
    INFILE DATAIN1;
    INPUT BOD $3-11 FITLVL 13 LGRD $15-16 LEXP $18 HGRD $20-21 HEXP $23
        PMOS $25-28             AMOS1 $39-42 AMOS2 $44-47 LDO $49
        SEX $51 MAC $72-79 T1 $30 T2 $31 T3 $32 T4 $33 T5 $34 T6 $35
        T7 $36 T8 $37;
    IF LGRD = 'O2' THEN LGRD = 2;
    ELSE IF LGRD = 'O3' THEN LGRD = 3;
    ELSE IF LGRD = 'WO' THEN LGRD = 1;
    ELSE IF LGRD = 'O4' THEN LGRD = 4;
    ELSE IF LGRD = 'O5' THEN LGRD = 5;
    IF HGRD = 'O2' THEN HGRD = 2;
    ELSE IF HGRD = 'O3' THEN HGRD = 3;
    ELSE IF HGRD = 'WO' THEN HGRD = 1;
    ELSE IF HGRD = 'O4' THEN HGRD = 4;
    ELSE IF HGRD = 'O5' THEN HGRD = 5;
    ELSE IF HGRD = '  ' THEN HGRD = LGRD;
    IF T1 = '*' THEN T1 = '1';
    ELSE IF T1 = ' ' THEN T1 = '0';
    IF T2 = '*' THEN T2 = '1';
    ELSE IF T2 = ' ' THEN T2 = '0';
    IF T3 = '*' THEN T3 = '1';
    ELSE IF T3 = ' ' THEN T3 = '0';
    IF T4 = '*' THEN T4 = '1';
    ELSE IF T4 = ' ' THEN T4 = '0';
    IF T5 = '*' THEN T5 = '1';
    ELSE IF T5 = ' ' THEN T5 = '0';
    IF T6 = '*' THEN T6 = '1';
    ELSE IF T6 = ' ' THEN T6 = '0';
    IF T7 = '*' THEN T7 = '1';
    ELSE IF T7 = ' ' THEN T7 = '0';
    IF T8 = '*' THEN T8 = '1';
    ELSE IF T8 = ' ' THEN T8 = '0';
    LENGTH OFFTYP $8;
    OFFTYP = T1 || T2 || T3 || T4 || T5 || T6 || T7 || T8;
    IF LDO = '*' THEN LDO = '0';
    IF LDO = 'L' THEN LDO = '1';
    IF LDO = 'U' THEN LDO = '2';
    IF SEX = '*' THEN SEX = '0';
    IF SEX = 'M' THEN SEX = '1';
    IF SEX = 'F' THEN SEX = '2';
    E1NUM+1;
    IF BOD NE LAG(BOD) THEN BODNUMA + 1;
    FILE DATAOUT1;
    PUT BOD $4-12 FITLVL 14 LGRD $16-17 LEXP $19 HGRD $21-22 HEXP $24
        PMOS $26-29 OFFTYP $31-38 AMOS1 $40-43 AMOS2 $45-48 LDO $50
        SEX $52 E1NUM 54-57 BODNUMA 59-61 MAC $73-80 ;
DATA FILE2;
    INFILE DATAIN2;
    INPUT BMOS $1-4 MCC $17-19 SPL 30 BOD $34-42 CAPACITY 45-46;
    IF CAPACITY = . THEN DELETE;
    E2NUM + 1;
    IF BOD NE LAG(BOD) THEN BODNUMB +1;
    FILE DATAOUT2;
    PUT BMOS 1-4 MCC $17-19 SPL 30 BOD $34-42 CAPACITY 45-46
        E2NUM 60-63 BODNUMB 65-68;
PROC SORT DATA=FILE1;
    BY BOD;
PROC SORT DATA=FILE2;
    BY BOD;
DATA FILE3;
    MERGE FILE1 FILE2;
        BY BOD ;
    PROC SORT DATA=FILE3;
        BY PMOS;
*  NEXT, ADD ON THE MOS NUMBERS (MOSNUM) ASSIGNED IN THE  ;
*  FILE MOVR-EP ARRAY. THIS WILL BE USED IN THE MATCHING ROUTINE.  ;
DATA FILE4;
    INFILE DATAIN3;
        INPUT MOSNUM 1-3 PMOS $5-8;
PROC SORT DATA = FILE4;
    BY PMOS;
DATA FILE5;
    MERGE FILE3 FILE4;
        BY PMOS;
PROC SORT DATA=FILE5;
        BY E1NUM;
DATA FILE9;
    SET FILE5;
    IF CAPACITY EQ . THEN DELETE;
*    NOW CLEAN UP THE NORMAL OUTPUT;
        IF BODNUMB = . THEN BODNUMB = 0;
        IF E1NUM = . THEN E1NUM = 0;
        IF E2NUM = . THEN E2NUM = 0;
        IF MOSNUM = . THEN MOSNUM =0;
        IF SPL = . THEN SPL = 0;
        IF CAPACITY = . THEN CAPACITY = 0;
        IF PMOS = ' ' THEN PMOS = '   0';

*     THE NEXT TASK IS TO ADD THE COST CODE CENTERS TO THE MCC'S WHICH ;
* WILL APPEAR IN THE OUTPUT FILE.  THESE COST CODE CENTERS WILL BE USED;
* IN THE MATCHING ROUTINE TO REFERENCE AN ARRAY CONTAINING THE COSTS   ;
* ASSOCIATED WITH MOVING AN OFFICER OF SOME PARTICULAR RANK FROM HIS   ;
* PRESENT MCC (WHICH WILL APPEAR ON THE USMC MOVRSUP OR NMOVSUP FILE)  ;
* TO HIS PROPOSED FUTURE MCC ("FMCC" - WHICH IS PULLED FROM THIS FILE.);
* BECAUSE SOME OF THE MCC'S ON THE PRESENT COST CODE CENTER LIST       ;
* ARE NOT YET PROPERLY MATCHED WITH A COST CODE CENTER, WE WILL        ;
* ARBITRARILY ASSIGN THEM TO COST CODE CENTER NUMBER 29 - KANSAS CITY. ;
DATA DD1;
    INFILE DATAIN4;
    INPUT @7 MCC $CHAR3. @11 COSTCTR $CHAR2. @14 CCNAME $CHAR10.;
    IF COSTCTR = ' 0' THEN DO;
        COSTCTR = '29';
        CCNAME = '*WARNING!*';
    END;
PROC SORT DATA = DD1;
    BY MCC;
PROC SORT DATA = FILE6;
    BY MCC;
DATA DD2;
    MERGE DD1 FILE6;
    BY MCC;
    IF COSTCTR = ' ' THEN COSTCTR = '**';
PROC SORT DATA = DD2;
    BY E1NUM;
DATA _NULL_;
    SET DD2;
    FILE DATAOUT3;
    IF CAPACITY EQ . THEN DELETE;
    IF E1NUM EQ . THEN DELETE;
    IF E2NUM EQ . THEN DELETE;
    IF BODNUMA EQ . THEN DELETE;
    IF MOSNUM EQ . THEN DELETE;
    IF SPL EQ . THEN DELETE;
    IF FITLVL EQ . THEN DELETE;
    PUT LGRD $1-2 LEXP $3 HGRD $5-6 HEXP $7
        PMOS $9-12 OFFTYP $14-21 AMOS1 $23-26 AMOS2 $28-31 LDO $33
        SEX $35 MOSNUM 37-40 E1NUM 42-45 BODNUMA 47-50 E2NUM 52-55 COSTCTR
        $59-60 MCC $62-64 SPL 66 FITLVL 68 CAPACITY 70-71 MAC $73-80;
      /*
      NOTES ON INDICES
          BODNUMA AND BODNUMB NEED NOT BE THE SAME.  DURING THE ASSIGNMENT
OF THE E2NUM'S, THE E2 CARDS WERE LEFT IN THE SAME ORDER THEY APPEAR IN
THE DICTIONARY, AND SOMETIMES (THOUGH RARELY) THE SAME BOD MAY APPEAR IN
THE DICTIONARY SEPARATED BY ANOTHER BOD. (EG. 3060LTCL AND 3060LTCL*)
THE E1NUM'S, ON THE OTHER HAND, WILL ALWAYS BE IN SEQUENCE, SINCE THE
FILE IS SORTED ON THE E1NUMS.
        */
;
```

APPENDIX O
E2ASRE1B SAS

1. PROGRAM TO ATTACH SUBSTITUTION LISTS TO FIXED (NON-MOVER) DEMANDS

```
***********************************************************************
*                                                                     *
*       * * * *   PROGRAM NAME: E2ASRE1A SAS         * * * *          *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                 * * *   OVERVIEW AND PURPOSE   * * *                *
*    THIS PROGRAM ATTACHES THE E1 CARD INFORMATION AND COST CODE      *
* INDICES TO THE FILE CONTAINING THE Fixed ASR-E2 CARDS INFORMATION.  *
* IT IS IDENTICAL TO E2ASRE1A, EXCEPT FOR THE FILE DEFINITIONS.       *
* THE FIRST TASK IS ATTACHING THE E1 CARD SUBSTITUTION LISTS TO THE   *
* DEMAND USING THE BOD'S WHICH WERE ATTACHED IN ASRE2A AS A MEANS OF  *
* LINKING THE DEMAND TO THE SUBSTITUTION LISTS.  THIS RESULTS IN A    *
* FILE WHICH HAS THE DEMAND BROKEN INTO SPECIFIC BILLETS (BY THE E2   *
* CARDS) WITH A SET OF UP TO FIVE SUBSTITUTION CRITERIA ATTACHED TO   *
* EACH SPECIFIC DEMAND.                                               *
*    THE NEXT TASK PERFORMED IN THIS PROGRAM IS THE PLACING OF THE    *
* APPROPRIATE COST CODE CENTER INDEX NEXT TO EACH DEMAND.  THIS       *
* CORRESPONDS, ON THE DEMAND SIDE, TO THE COST CODE CENTER INDEX WHICH *
* WAS ATTACHED TO EACH INDIVIDUAL IN FREE-FIX SAS, ON THE SUPPLY SIDE. *
*    MOST OF THE FUNCTIONS AND TASKS PERFORMED IN THE PROGRAM ARE     *
* EXPLAINED AS THEY ARE PERFORMED, THUS THE BODY OF THE PROGRAM IS    *
* SELF-DOCUMENTING.                                                   *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                ***   DEFINITION OF TERMS   ***                      *
*                                                                     *
*   AMOS1      FIRST AMOS APPEARING ON THE E1 CARD (SUBSTITUTION)     *
*   AMOS2      SECOND AMOS APPEARING ON THE E1 CARD (SUBSTITUTION)    *
*   BOD        BILLET OFFICER DESCRIPTION                             *
*   BODNUMA    INDEX ON THE BOD FOUND IN THE E1 CARDS                 *
*   BODNUMB    INDEX ON THE BOD FOUND IN THE E2 CARDS                 *
*   CAPACITY   TOTAL AUTHORIZED DEMAND AT THE CURRENT MCC             *
*   COSTCTR    COST CENTER CODE INDEX                                 *
*   E1NUM      INDEX ON THE E1 CARDS                                  *
*   E2NUM      INDEX ON THE E2 CARDS                                  *
*   FITLVL     FIT LEVEL DEFINED BY ORDER OF SUBSTITUTION PREFERENCE  *
*   GD         GRADE OR RANK (INTEGER)                                *
*   HEXP       EXPERIENCE CODE ASSOCIATED WITH HIGH GRADE             *
*   HGRD       HIGHEST ACCEPTABLE GRADE FOR A SUBSTITUTION            *
*   LDO        DUTY RESTRICTION CODE                                  *
*   LEXP       EXPERIENCE CODE ASSOCIATED WITH LOW GRADE              *
*   LGRD       LOWEST ACCEPTABLE GRADE FOR A SUBSTITUTION             *
*   MAC        MONITORED ACTIVITY CODE                                *
*   MCC        CURRENT MONITORED COMMAND CODE (LOCATION)              *
*   MCC2       NEXT MONITORED COMMAND CODE (LOCATION)                 *
*   NUM        NUMBER OF MARINES FILLING THE DEMAND BEING LOOKED AT   *
*   OT         OFFICER TYPE                                           *
*   PMOS       PRIMARY MOS APPEARING ON AN E1 CARD (SUBSTITUTION)     *
*   SA         SPLIT ADJUSTMENT - ADD (A), SUBTRACT (S), OR PROPORT (P) *
*   SAMT       SPLIT AMOUNT   NAME CONTAINING START & ENDING POINTS   *
*   SEX        SEX CODE RESTRICTION INDICATOR                         *
*   SPL        STAFFING PRECEDENCE LEVEL                              *
*   T1 - T9    0/1 FLAGS FOR OFFICER CODE TYPES FOUND IN E1 CARDS     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *    FILE DEFINITION   *  *  *  *  *  *  *  *  *  *
```

```
CMS FILEDEF DATAIN1 DISK WKE1 CRD A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN2 DISK E2ASR MIX-NMOV A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN3 DISK NMOVR-EP ARRAY A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAIN4 DISK CCC-MCC CONVERT A1;
CMS FILEDEF DATAOUT1 DISK E1-INDEX CRD-B A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT2 DISK E2-INDEX CRD-B A (RECFM F LRECL 80 BLOCK 80;
CMS FILEDEF DATAOUT3 DISK NMOV-DEM INPUT A (RECFM F LRECL 80 BLOCK 80;
***********************************************************************
OPTIONS LINESIZE=80;
    /*
      THE FIRST PORTION OF THIS SAS JOB PUTS 2 INDICES ON THE E1 CARDS.
THE FIRST INDEX SIMPLY IS A COUNT OF THE E1 CARDS AND CAN BE USED TO
PUT THE CARDS BACK IN ORDER AFTER ANY COMBINATION OF OTHER SORTS ARE
PERFORMED ON THE FILE.  THE SECOND INDEX MARKS ALL THOSE E1 CARDS WHICH
HAVE THE SAME BOD.  THIS INDEX CAN BE USED TO RESTORE THE ORDER OF THE
E1 CARDS BY BOD AFTER THE FILE HAS BEEN SORTED SOME OTHER WAY.
      */;
DATA FILE1;
    INFILE DATAIN1;
    INPUT BOD $3-11 FITLVL 13 LGRD $15-16 LEXP $18 HGRD $20-21 HEXP $23
        PMOS $25-28              AMOS1 $39-42 AMOS2 $44-47 LDO $49
        SEX $51 MAC $72-79 T1 $30 T2 $31 T3 $32 T4 $33 T5 $34 T6 $35
        T7 $36 T8 $37;
    IF LGRD = '02' THEN LGRD = 2;
    ELSE IF LGRD = '03' THEN LGRD = 3;
    ELSE IF LGRD = 'WO' THEN LGRD = 1;
    ELSE IF LGRD = '04' THEN LGRD = 4;
    ELSE IF LGRD = '05' THEN LGRD = 5;
    IF HGRD = '02' THEN HGRD = 2;
    ELSE IF HGRD = '03' THEN HGRD = 3;
    ELSE IF HGRD = 'WO' THEN HGRD = 1;
    ELSE IF HGRD = '04' THEN HGRD = 4;
    ELSE IF HGRD = '05' THEN HGRD = 5;
    ELSE IF HGRD = '  ' THEN HGRD = LGRD;
    IF T1 = '*' THEN T1 = '1';
    ELSE IF T1 = ' ' THEN T1 = '0';
    IF T2 = '*' THEN T2 = '1';
    ELSE IF T2 = ' ' THEN T2 = '0';
    IF T3 = '*' THEN T3 = '1';
    ELSE IF T3 = ' ' THEN T3 = '0';
    IF T4 = '*' THEN T4 = '1';
    ELSE IF T4 = ' ' THEN T4 = '0';
    IF T5 = '*' THEN T5 = '1';
    ELSE IF T5 = ' ' THEN T5 = '0';
    IF T6 = '*' THEN T6 = '1';
    ELSE IF T6 = ' ' THEN T6 = '0';
    IF T7 = '*' THEN T7 = '1';
    ELSE IF T7 = ' ' THEN T7 = '0';
    IF T8 = '*' THEN T8 = '1';
    ELSE IF T8 = ' ' THEN T8 = '0';
    LENGTH OFFTYP $8;
    OFFTYP = T1 || T2 || T3 || T4 || T5 || T6 || T7 || T8;
    IF LDO = '*' THEN LDO = '0';
    IF LDO = 'L' THEN LDO = '1';
    IF LDO = 'U' THEN LDO = '2';
    IF SEX = '*' THEN SEX = '0',
    IF SEX = 'M' THEN SEX = '1';
    IF SEX = 'F' THEN SEX = '2';
    E1NUM+1;
    IF BOD NE LAG(BOD) THEN BODNUMA + 1;
    FILE DATAOUT1;
    PUT BOD $4-12 FITLVL 14 LGRD $16-17 LEXP $19 HGRD $21-22 HEXP $24
        PMOS $26-29 OFFTYP $31-38 AMOS1 $40-43 AMOS2 $45-48 LDO $50
        SEX $52 E1NUM 54-57 BODNUMA 59-61 MAC $73-80 ;
DATA FILE2;
    INFILE DATAIN2;
    INPUT BMOS $1-4 MCC $17-19 SPL 30 BOD $34-42 CAPACITY 45-46;
    IF CAPACITY = . THEN DELETE;
    E2NUM + 1;
    IF BOD NE LAG(BOD) THEN BODNUMB +1;
    FILE DATAOUT2;
    PUT BMOS 1-4 MCC $17-19 SPL 30 BOD $34-42 CAPACITY 45-46
        E2NUM 60-63 BODNUMB 65-68;
PROC SORT DATA=FILE1;
    BY BOD;
PROC SORT DATA=FILE2;
    BY BOD;
DATA FILE3;
    MERGE FILE1 FILE2;
        BY BOD ;
    PROC SORT DATA=FILE3;
        BY PMOS;
*  NEXT, ADD ON THE MOS NUMBERS (MOSNUM) ASSIGNED IN THE  ;
*  FILE NMOVR-EP ARRAY. THIS WILL BE USED IN THE MATCHING ROUTINE.  ;
DATA FILE4;
    INFILE DATAIN3;
        INPUT MOSNUM 1-3 PMOS $5-8;
PROC SORT DATA = FILE4;
    BY PMOS;
DATA FILE5;
    MERGE FILE3 FILE4;
        BY PMOS;
PROC SORT DATA=FILE5;
        BY E1NUM;
DATA FILE9;
    SET FILE5;
    IF CAPACITY EQ . THEN DELETE;
*   NOW CLEAN UP THE NORMAL OUTPUT;
        IF BODNUMB = . THEN BODNUMB = 0;
        IF E1NUM = . THEN E1NUM = 0;
        IF E2NUM = . THEN E2NUM = 0;
        IF MOSNUM = . THEN MOSNUM =0;
        IF SPL = . THEN SPL = 0;
        IF CAPACITY = . THEN CAPACITY = 0;
        IF PMOS = ' ' THEN PMOS = '   0';

*     THE NEXT TASK IS TO ADD THE COST CODE CENTERS TO THE MCC'S WHICH ;
* WILL APPEAR IN THE OUTPUT FILE.  THESE COST CODE CENTERS WILL BE USED;
* IN THE MATCHING ROUTINE TO REFERENCE AN ARRAY CONTAINING THE COSTS   ;
* ASSOCIATED WITH MOVING AN OFFICER OF SOME PARTICULAR RANK FROM HIS   ;
* PRESENT MCC (WHICH WILL APPEAR ON THE USMC MOVRSUP OR NMOVSUP FILE)  ;
* TO HIS PROPOSED FUTURE MCC ("FMCC" - WHICH IS PULLED FROM THIS FILE.);
* BECAUSE SOME OF THE MCC'S ON THE PRESENT COST CODE CENTER LIST       ;
* ARE NOT YET PROPERLY MATCHED WITH A COST CODE CENTER, WE WILL        ;
* ARBITRARILY ASSIGN THEM TO COST CODE CENTER NUMBER 29 - KANSAS CITY. ;
DATA DD1;
    INFILE DATAIN4;
    INPUT @7 MCC $CHAR3. @11 COSTCTR $CHAR2. @14 CCNAME $CHAR10.;
    IF COSTCTR = ' 0' THEN DO;
        COSTCTR = '29';
        CCNAME = '*WARNING!*';
    END;
PROC SORT DATA = DD1;
    BY MCC;
PROC SORT DATA = FILE6;
    BY MCC;
DATA DD2;
    MERGE DD1 FILE6;
    BY MCC;
    IF COSTCTR = ' ' THEN COSTCTR = '**';
PROC SORT DATA = DD2;
    BY E1NUM;
DATA _NULL_;
    SET DD2;
```

```
        FILE DATAOUT3;
        IF CAPACITY EQ . THEN DELETE;
        IF E1NUM EQ . THEN DELETE;
        IF E2NUM EQ . THEN DELETE;
        IF BODNUMA EQ . THEN DELETE;
        IF MOSNUM EQ . THEN DELETE;
        IF SPL EQ . THEN DELETE;
        IF FITLVL EQ . THEN DELETE;
        PUT LGRD $1-2 LEXP $3 HGRD $5-6 HEXP $7
            PMOS $9-12 OFFTYP $14-21 AMOS1 $23-26 AMOS2 $28-31 LDO $33
            SEX $35 MOSNUM 37-40 E1NUM 42-45 BODNUMA 47-50 E2NUM 52-55 COSTCTR
            $59-60 MCC $62-64 SPL 66 FITLVL 68 CAPACITY 70-71 MAC $73-80;
          /*
          NOTES ON INDICES
              BODNUMA AND BODNUMB NEED NOT BE THE SAME.  DURING THE ASSIGNMENT
    OF THE E2NUM'S, THE E2 CARDS WERE LEFT IN THE SAME ORDER THEY APPEAR IN
    THE DICTIONARY, AND SOMETIMES (THOUGH RARELY) THE SAME BOD MAY APPEAR IN
    THE DICTIONARY SEPARATED BY ANOTHER BOD. (EG. 3060LTCL AND 3060LTCL*)
    THE E1NUM'S, ON THE OTHER HAND, WILL ALWAYS BE IN SEQUENCE, SINCE THE
    FILE IS SORTED ON THE E1NUMS.
          */
    ;
```

APPENDIX P
MATCH-AL FORTRAN

1. PROGRAM TO MATCH PEOPLE TO JOBS

```
************************************************************************
*                                                                      *
*       * * * *   PROGRAM NAME: MATCH-AL FORTRAN    * * * *            *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                  * * *   OVERVIEW AND PURPOSE   * * *                *
*                                                                      *
*     IN THIS PROGRAM THE INVENTORY IS MATCHED TO THE DEMAND DEFINED   *
* BY THE ADJUSTED ASR.  IN ORDER TO PERFORM THE MATCH, NUMEROUS FILES  *
* WILL BE USED.  THE FILE E1E2-MRG CRD CONTAINS ALL OF THE E1, E2 AND  *
* ASR INFORMATION ROLLED UP IN A CONVOLUTED BUT COMPACT FORM.  THE     *
* INFORMATION ON THE INVENTORY OF THE MARINE CORPS (IE. THE SUPPLY OF  *
* OFFICERS) WHICH IS NECESSARY FOR THE MATCHING PROCESS IS FOUND IN    *
* THE FILE WORKING INVENTRY.   BUT BECAUSE ONLY ABOUT A THIRD OF THE   *
* MARINE CORPS IS ELIGIBLE TO MOVE IN ANY YEAR (WHICH IS THE PERIOD    *
* CONSIDERED IN THE PRESENT ALLOCATION MODEL), THE MATCHING WILL BE    *
* PERFORMED IN TWO PARTS.  IN THE FIRST PART, ALL LEGAL MATCHES FOR    *
* THOSE MARINES WHO WILL BE MOVING DURING THE NEXT TIME PERIOD (OR     *
* WINDOW) ARE MADE.   IN THIS PART OF THE PROGRAM, ALL INDIVIDUALS ARE *
* MATCHED TO SUBSTITUTION CRITERIA THEY CAN FILL.  THESE PEOPLE-TO-JOB-*
* TYPE MATCHES ARE THEN MATCHED TO SPECIFIC DEMANDS.   THE SECOND      *
* PART OF THE PROGRAM,MATCHES ALL "NON-MOVERS (FIXED PERSONNEL) TO     *
* OTHER JOBS WITHIN THEIR MCC WHICH THEY MIGHT BE ABLE TO FILL AT A    *
* BETTER "FIT" ( DEFINED BY THE E1 CARDS.)  THE ARCS GENERATED WITHIN  *
* UNITS BY THIS OTHER PROGRAM ARE THEN COMBINED WITH THE MOVERS' ARCS, *
* AND ARE ALL FORMATTED FOR INPUT INTO GNET.  DIVIDING THE MATCHING    *
* PROCESS INTO THESE TWO PHASES AND LIMITING THE ARCS FOR MOVERS TO    *
* JOBS WHICH ARE NOT OCCUPIED BY NON-MOVERS REDUCES THE NUMBER OF ARCS *
* WHICH MUST BE GENERATED.                                             *
*                                                                      *
*     THIS PROGRAM IS BROKEN DOWN INTO SEVERAL STAGES.  FIRST, A COST  *
* MATRIX IS READ INTO MEMORY , BASED ON DISTANCE PLUS AN ARBITRARY     *
* CONSTANT(2000), AND INDEXED ON COST CENTER CODES.  NEXT THE ENTIRE   *
* POPULATION OF MOVERS IS READ INTO MEMORY, ALREADY SORTED BY MOS AND  *
* GRADE.  ALSO READ IN IS A POINTER ARRAY WHICH WILL BE USED TO SPEED  *
* UP THE MATCHING PROCESS BY DIRECTING THE SEARCH FOR THOSE WHO FILL   *
* THE REQUIREMENT FOR A PARTICULAR SUBSTITUTION CRITERIA TO THE EXACT  *
* MOS AND GRADE INVOLVED.                                              *
*     NEXT, EACH E1E2-MRG CARD DEFINING THE DEMAND IS READ IN.  EVERY  *
* INDIVIDUAL  IN THE INVENTORY WHO MATCHES THE ACCEPTABLE              *
* SUBSTITUTION CRITERIA FOR THAT DEMAND IS FOUND BY SEARCHING THROUGH  *
* THE INVENTORY  USING THE POINTER ARRAYS.                              *
*     THE ENTIRE PROCESS IS THEN REPEATED FOR NON-MOVERS.              *
*     THE RESULTING COMBINED FILE CONTAINS ALL PEOPLE TO BILLET        *
* MATCHES WHICH WILL BE USED TO GENERATE THE NETWORK ONCE THEY ARE     *
* SORTED IN THE NEXT PROGRAM, BIGSORT SAS.                             *
*     EACH STEP OF THE PROCESS IS DOCUMENTED IN MORE DETAIL IN THE     *
* PROGRAM BELOW.                                                       *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                  ***   FILE DEFINITIONS   ***                        *
*                                                                      *
*    FILEDEF     FILE IDENTIFICATION                PURPOSE            *
*       1          ALPHA DATA           FILE WITH ALPHA VALUE (FROM USER)*
*       8          USMC MOVRSUP         CONTAINS INVENTORY OF MOVERS     *
*      10          USMC NONMSUP         CONTAINS INVENTORY OF NON-MOVERS *
*        12      MOVR-EP ARRAY        POINTER ARRAY TO MOVER MOS/GRADE *
*        13      NMOVR-EP ARRAY       POINTER ARRAY FOR NON-MOVERS     *
*        21      MOVR-DEM INPUT       CONTAINS COMBINED E1-E2-ASR INFO *
*                                     FOR MOVERS                       *
*        22      DEBUG OUTPUT         PROVIDES INFORMATION ON EACH     *
*                                     STAGE OF THE PROGRAM TO THE USER *
*        23      EASY-MOS MATCH       CONTAINS MOS/GRADE E1 CARDS      *
*        24      HARD-MOS MATCH       CONTAINS OCC FIELD/AMOS E1 CARDS *
*        25      NON-MOS MATCH        CONTAINS "OFFICER TYPE" E1 CARDS *
*        26      MATCH OUTPUT         CONTAINS PEOPLE-TO-JOB MATCHES   *
*        27      NMOV-DEM INPUT       CONTAINS COMBINED E1-E2-ASR INFO *
*                                     FOR NON-MOVERS                   *
*        28      COST-CTR DIST-MAT    MATRIX OF DIST BETWEEN COST CTRS *
*        29      TEST-OF COST-OUT     ENSURES PROPER READING OF FILE28 *
*        30      SUP-SIZE DATA        FILE WITH TOTAL NO. IN INVENTORY *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                  ***   DEFINITION OF TERMS   ***                     *
*                                                                      *
*    ALPHA      COEFFICIENT USED TO DETERMINE THE RELATIVE WEIGHTS OF  *
*               THE FIT AND PCS COST OBJECTIVES                        *
*    AMOS1      FIRST AMOS ASSOCIATED WITH THE SUBSTITUTION CRITERIA   *
*    AMOS2      SECOND AMOS ASSOCIATED WITH THE SUBSTITUTION CRITERIA  *
*    BEGSCH     POINTER INDEX TO SORTED INVENTORY; WHERE TO BEGIN SEARCH *
*    BODE2      INDEX ON THE SPECIFIC DEMAND ATTACHED TO A BOD         *
*    BODNUM     INDEX ON THE BOD ASSOCIATED WITH A SUBSTITUTION        *
*    CAPCTY     THE DEMAND (FLOW CAPACITY) AT A PARTICULAR DEMAND      *
*    COST       DISTANCE BETWEEN PMCC AND FMCC USED AS A BASIC PCS COST *
*    CSTCTR     COST CENTER CODE OF A DEMAND                           *
*    E1NUM      THE E1 INDEX FOR A PARTICULAR E1-E2-ASR CARD           *
*    E2NUM      THE E2 INDEX FOR A PARTICULAR MATCH                    *
*    ENDGRP     MARKS THE END OF A GROUP OF E1-E2-ASR CARDS BEING INPUT *
*    ENDSCH     POINTER INDEX TO SORTED INVENTORY; WHERE TO END SEARCH *
*    EXPEXC     EXPERIENCE EXCEPTION - TELLS IF MAY IGNORE E1 EXP CODE *
*    FITLVL     FIT LEVEL FOR A PARTICULAR SUBSTITUTION                *
*    FMCC       FUTURE MCC - MCC ASSOCIATED WITH THE DEMAND            *
*    HEXP       EXPERIENCE FLAF FOR HGRD                               *
*    HGRD       HIGHEST ACCEPTABLE GRADE FOR A PARTICULAR SUBSTITUTION *
*    IAMOS1     FIRST AMOS OF THE INDIVIDUAL BEING CONSIDERED          *
*    IAMOS2     SECOND AMOS OF THE INDIVIDUAL BEING CONSIDERED         *
*    ICOST      THE COST ATTACHED TO A PARTICULAR MATCH                *
*    ICOSTC     COST CODE CENTER OF THE INDIVIDUAL BEING CONSIDERED    *
*    IDNUM      INDEX FOR EACH INDIVIDUAL WITHIN THE INVENTORY         *
*    IEXP       EXPERIENCE CODE OF THE INDIVIDUAL BEING CONSIDERED     *
*    IGRD       GRADE OF THE INDIVIDUAL BEING CONSIDERED FOR A MATCH   *
*    ILDO       DUTY RESTRICTION OF THE INDIVIDUAL BEING CONSIDERED    *
*    IMOS       PMOS OF THE INDIVIDUAL BEING CONSIDERED                *
*    IMOSNUM    INDEX OF INDIVIDUAL'S PMOS                             *
*    ISEX       SEX OF THE INDIVIDUAL BEING CONSIDERED                 *
*    LAST1M     LAST CHARACTER IS MCC                                  *
*    LDO        DUTY RESTRICTION CODE FOR THE SUBSTITUTION CRITERIA    *
*    LEXP       EXPERIENCE FLAG FOR LGRD                               *
*    LGRD       LOWEST ACCEPTABLE GRADE FOR A PARTICULAR SUBSTITUTION  *
*    LSTBOD     INDICATES THE LST BOD THE INDIVIDUAL WAS MATCHED TO -  *
*    MARKR1     MARKER FOR BEGINNING OF EACH SET OF E1 CARDS IN A GROUP *
*    MARKR2     MARKER FOR BEGINNING OF EACH SET OF E2 CARDS IN A GROUP *
*    MOS        TEMPORARY VARIABLE TO STORE PMOS IN MATCHING ROUTINE   *
*    MOSNUM     THE MOS INDEX NUMBER FOR A PARTICULAR PMOS             *
*    NUMFIT     TELLS THE NUMBER OF FIT LEVELS IN A GIVEN BOD          *
*    NUMMCC     TELLS THE NUMBER OF DIFFERENT MCC'S WITH A GIVEN BOD   *
*    OFFTYP     OFFICER TYPE                                           *
*    PMCC       PRESENT MCC OF THE INDIVIDUAL BEING CONSIDERED         *
*    PMOS       PRIMARY MOS ASSOCIATED WITH THE SUBSTITUTION CRITERIA  *
*    SEX        SEX CODE APPEARING WITH THE SUBSTITUTION CRITERIA      *
*    SPL        STAFFING PRECEDENCE LEVEL FOR A BILLET                 *
*    SUPSIZ     THE TOTAL NUMBER OF PEOPLE ON THE INVENTORY LIST       *
*    TAMOS1     TEMPORARY VARIABLE TO HOLD THE VALUE OF AMOS1          *
*    TAMOS2     TEMPORARY VARIABLE TO HOLD THE VALUE OF AMOS2          *
*    TBODA      TEMPORARY VARIABLE TO HOLD THE VALUE OF BODNUM         *
*    TCAP       TEMPORARY VARIABLE TO HOLD THE VALUE OF CAPCTY          *
*    TCCTR      TEMPORARY VARIABLE TO HOLD THE VALUE OF CSTCTR          *
*    TE1NUM     TEMPORARY VARIABLE TO HOLD THE VALUE OF E1NUM           *
*    TE2NUM     TEMPORARY VARIABLE TO HOLD THE VALUE OF E2NUM           *
*    TFITLV     TEMPORARY VARIABLE TO HOLD THE VALUE OF FITLVL          *
*    TFMCC      TEMPORARY VARIABLE TO HOLD THE VALUE OF FMCC            *
*    THEXP      TEMPORARY VARIABLE TO HOLD THE VALUE OF HEXP            *
*    THGRD      TEMPORARY VARIABLE TO HOLD THE VALUE OF HGRD            *
*    TKOUNT     COUNTER OF TOTAL NUMBER OF E1-E2-ASR GROUPS             *
*    TLDO       TEMPORARY VARIABLE TO HOLD THE VALUE OF LDO             *
*    TLEXP      TEMPORARY VARIABLE TO HOLD THE VALUE OF LEXP            *
*    TLGRD      TEMPORARY VARIABLE TO HOLD THE VALUE OF LGRD            *
*    TMOSNO     TEMPORARY VARIABLE TO HOLD THE VALUE OF MOSNUM          *
*    TOFTYP     TEMPORARY VARIABLE TO HOLD THE VALUE OF OFFTYP          *
*    TOTFIT     TOTAL NUMBER OF E1 CARDS READ IN EACH GROUP OF          *
*               E1-E2-ASR CARDS                                         *
*    TPMOS      TEMPORARY VARIABLE TO HOLD THE VALUE OF PMOS            *
*    TSEX       TEMPORARY VARIABLE TO HOLD THE VALUE OF SEX             *
*    TSPL       TEMPORARY VARIABLE TO HOLD THE VALUE OF SPL             *
*                                                                       *
*************************************************************************
* * * * * *     DECLARE, DIMENSION, AND INITIALIZE     * * * * * * *
       CHARACTER*1 LAST1M,LEXP(500),HEXP(500),TLEXP,THEXP
       CHARACTER*2 EXPEXC
       CHARACTER*4 FMCC(500),PMOS(500),AMOS1(500),
      CAMOS2(500),TPMOS,TAMOS1,TAMOS2,TFMCC,MOS,IMOS(20000),IAMOS1(20000)
      C,IAMOS2(20000),PMCC(20000)
       CHARACTER*8 OFFTYP(500),TOFTYP
       INTEGER*2 IEXP(20000),CSTCTR(500),ICOSTC(20000),TCCTR
       INTEGER*2 MARKR1,MARKR2,NUMFIT,LGRD(500),HGRD(500),TLGRD,THGRD,
      CIGRD(20000),SEX(500),TSEX,ISEX(20000),LDO(500),TLDO,ILDO(20000),
      CSPL(500),FITLVL(500)
       INTEGER*4 ENDGRP,NUMMCC,E1NUM(500),MOSNUM(500),LSTBOD(20000),
      CBODNUM(500),BODE2(500),E2NUM(500),CAPCTY(500)
      C,TE1NUM,TE2NUM,TSPL,TFITLV,TMOSNO,TOTFIT,
      CTCAP,TBODA,TKOUNT,IMOSNO(20000)
       INTEGER IDNUM(20000),BEGSCH(250,5),ENDSCH(250,5), COST(63,63)
      C,ICOST,SUPSIZ
       REAL ALPHA
       DIMENSION BAQ(2,9),MILES(8,8),
*     CALL SETIME
********   READ COST CENTER PCS COST ARRAY INTO RESIDENT MEMORY   ******
       DO 12 I = 1,63
          DO 12 J = 1,63
             READ(15,121) COST(I,J)
             COST(I,J) = COST(I,J) + 2000
             IF (COST(I,J) .LE. 2075) COST(I,J) = 0
12    CONTINUE
       DO 13 I = 1,63
          DO 13 J = 1,63
             WRITE (16,121) COST(I,J)
13    CONTINUE
121   FORMAT(I4)

************   READ MOVER INVENTORY INTO RESIDENT MEMORY   *************
       K = 1
       DO 5 I = 1,20000
         READ(08,102,END=9991) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
      CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
102     FORMAT(A4,2X,I1,1X,A4,1X,A4,1X,A3,1X,I1,1X,I1,1X,I1,3X,I5,1X,I3,
      C39X,I2)
       LSTBOD(I) = 0
       IF (K .EQ. 1000) THEN
        WRITE(02,*) 'K =',K
        WRITE(02,102) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
      CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
       ELSE IF (K .EQ. 2000) THEN
        WRITE(02,*) 'K =',K
        WRITE(02,102) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
      CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
       ENDIF
       K=K+1


5     CONTINUE
9991  CONTINUE
       WRITE(02,*) 'THERE ARE',K,'PEOPLE IN THE MOVER INVENTORY FILE.'
       SUPSIZ = K
**************   READ EP ARRAY INTO RESIDENT MEMORY   *****************
*     NEXT, READ IN THE ENTRY POINT ARRAY DEVELOPED FOR THE
* MOVERS, CALLED MOVR-EP ARRAY
*     IT IS IMPORTANT TO REMEMBER THAT THIS ARRAY REFERS TO THE PMOS'S
* OF THE MARINES ON THE INVENTORY, AND NOT THE BMOS'S ON THE E1 CARDS.
* CONSEQUENTLY, IF A BMOS APPEARS ON THE E1 CARDS WHICH IS NOT
* POSSESSED BY ANY MARINES, THIS PROGRAM WILL HAVE PROBLEMS MATCHING,
* AND THE INDICES WILL PROBABLY GET HOPELESSLY MESSED UP IN THE MATCH
* ROUTINE.  THIS IN FACT HAPPENED IN THE FIRST RUN OF THIS PROGRAM.
* TWO MOS'S, 7580 AND 7584, WHICH APPEARED IN THE E1 CARDS AS BMOS'S,
* DID NOT APPEAR AS PMOS'S ANYWHERE IN THE INVENTORY, AND HENCE, DID
* NOT HAVE PROPER INDICES FOR THE SEARCH.  (EG. SEE LINE 311 WHERE THE
* INDEX "L" IS SET.)  THIS DOES NOT RESULT IN ANY ERROR STATEMENT,
* BUT CAUSES UNCERTAIN RESULTS.  IN THE PROTOTYPE, THE E1 CARDS WITH THE
* OFFENDING BMOS'S WERE REMOVED.
       DO 6 I = 1,250
          DO 6 J = 1,5
             READ(09,103,END=9992) BEGSCH(I,J),ENDSCH(I,J)
103   FORMAT(12X,I5,1X,I5)
* * * * * * * *   BEGIN TEST PRINTOUT
       WRITE(02,*) I,J,'BEGSCH(I,J)=',BEGSCH(I,J),'ENDSCH(I,J)=',
      CENDSCH(I,J)
* * * * * * * *   END TEST PRINTOUT
6     CONTINUE
9992  CONTINUE
***********************************************************************
*     READ IN EACH LINE FROM THE FILE MOVR-DEM INPUT FOR MATCHING     *
*     INITIALIZE
       I = 1
       TOTFIT = 0
       READ(01,101) LGRD(I),LEXP(I),HGRD(I),HEXP(I),PMOS(I),OFFTYP(I),
      CAMOS1(I),AMOS2(I),LDO(I),SEX(I),MOSNUM(I),E1NUM(I),BODNUM(I),
      CE2NUM(I),CSTCTR(I),FMCC(I),SPL(I),FITLVL(I),CAPCTY(I)
101   FORMAT (I2,A1,1X,I2,A1,1X,A4,1X,A8,1X,A4,1X,A4,1X,I1,
      C1X,I1,1X,I4,1X,I4,1X,I4,1X,I4,3X,I2,1X,A3,1X,I1,1X,I1,1X,I2)
       TKOUNT = 1
501   I = I+1
       READ(01,101,END=9993) LGRD(I),LEXP(I),HGRD(I),HEXP(I),PMOS(I),
      COFFTYP(I),AMOS1(I),AMOS2(I),LDO(I),SEX(I),MOSNUM(I),E1NUM(I),
      CBODNUM(I),E2NUM(I),CSTCTR(I),FMCC(I),SPL(I),FITLVL(I),CAPCTY(I)
       IF (BODNUM(I) .EQ. BODNUM(I-1)) GOTO 501
       IF (BODNUM(I) .NE. BODNUM(I-1)) THEN
          ENDGRP = I-1
*     STORE THE FIRST VARIABLE IN THE NEXT BOD SET IN TEMPORARY VALUES
          TLGRD = LGRD(I)
          TLEXP = LEXP(I)
          THGRD = HGRD(I)
          THEXP = HEXP(I)
          TPMOS = PMOS(I)
          TOFTYP = OFFTYP(I)
          TAMOS1 = AMOS1(I)
          TAMOS2 = AMOS2(I)
          TLDO = LDO(I)
          TSEX = SEX(I)
          TMOSNO = MOSNUM(I)
          TE1NUM =  E1NUM(I)
          TBODA = BODNUM(I)
```

```
            TCCTR = CSTCTR(I)
            TE2NUM = E2NUM(I)
            TFMCC = FMCC(I)
            TSPL = SPL(I)
            TFITLV = FITLVL(I)
            TCAP = CAPCTY(I)
         ENDIF
* * * * * * * * *   BEGIN TEST PRINTOUT
         WRITE(02,*) '**************************************************'
         WRITE(02,*) 'GROUP ',TKOUNT,' HAS ',ENDGRP,' CARDS.'
* * * * * * * * *   END TEST PRINTOUT
*     EACH OF THE GROUPS WE HAVE JUST READ IN CONTAIN ALL OF THE E1-CARD
*  AND ASR INFORMATION FOR THE MOVERS.  EACH GROUP CONTAINS ALL OF THE
*  E1-CARD DATA AND THE MCC AND DEMAND (OR CAPACITY) FOR EACH DEMAND
*  ASSOCIATED WITH THAT E1 INFORMATION.  BECAUSE WE HAVE COMPRESSED SO
*  MUCH INFORMATION INTO A SINGLE FILE, WE MUST NOW DETERMINE TWO THINGS
*  ABOUT EACH GROUP.  FIRST, WE MUST FIND THE NUMBER OF SUBSTITUTION
*  LEVELS FOR THE DEMANDS OF THIS TYPE (ALSO CALLED THE "BOD SET").
*  ADDITIONALLY, WE MUST ASCERTAIN THE NUMBER OF DEMANDS IN THE GROUP
*  WHICH CORRESPOND TO THE BOD SET.  WE WILL USE THESE TWO ITEMS OF
*  INFORMATION AS INDICES TO HELP US PERFORM THE MATCHING OF PEOPLE TO
*  BILLETS WITH SOME SEMBLANCE OF EFFICIENCY. THE NUMBER OF SUBSTITUTION
*  LEVELS, CALLED "NUMFIT", TELLS US WHEN TO BEGIN SEARCHING THROUGH THE
*  NEXT BOD.  THE NUMBER OF DEMAND LOCATIONS, WHICH IS EQUIVALENT TO THE
*  NUMBER OF MCC'S, IS CALLED "NUMMCC".  THIS INDEX TELLS US HOW MANY
*  (AND WHICH) MCC'S WILL BE MATCHED TO THE VALID PEOPLE-SUBSTITUTION
*  MATCHES WE FIND BY MATCHING THE BOD SETS TO THE INVENTORY.
         MARKR1 = 0
         MARKR2 = 0
         IF (ENDGRP .EQ. 1) THEN
            NUMFIT = 1
            NUMMCC = 1
         ELSE IF (ENDGRP .GT. 1) THEN
            DO 10 I = 2,ENDGRP
               IF ((FITLVL(I) .EQ. FITLVL(I-1)) .AND. (MARKR1 .EQ. 0)) THEN
                  NUMFIT = I-1
                  MARKR1 = 1
               ELSE IF ((FITLVL(I) .GT. FITLVL(I-1)) .AND. (I .EQ. ENDGRP))
        CTHEN
                  NUMFIT = I
               ENDIF
               IF ((FMCC(I) .EQ. FMCC(I-1)) .AND. (MARKR2 .EQ. 0)) THEN
                  NUMMCC = I-1
                  MARKR2 = 1
               ELSE IF ((FMCC(I) .NE. FMCC(I-1)) .AND. (I .EQ. ENDGRP)) THEN
                  NUMMCC = I
               ENDIF
10        CONTINUE
            ENDIF
*     AT THIS POINT, WE SHOULD HAVE THE NUMBER OF FIT LEVELS AND DEMANDS
*   WITHIN THE GROUP. AS A CHECK, WE WILL PRINT OUT THE RESULTS.
*   WE WILL INCLUDE A RUNNING TOTAL OF THE TOTAL NUMBER OF E1-CARDS
*   INCLUDED IN THE FILE, AND CALL THIS VARIABLE TOTFIT.
         TOTFIT = TOTFIT + NUMFIT
* * * * * * * * *   BEGIN TEST PRINTOUT
         WRITE(02,*) 'THE NUMBER OF FIT LEVELS IN GROUP ',TKOUNT,' IS ',
        CNUMFIT,'.'
         WRITE(02,*) 'SO FAR THERE HAVE BEEN A TOTAL OF',TOTFIT,'E1 CARDS.'
         WRITE(02,*) 'THE NUMBER OF MCC DEMANDS IN GROUP ',TKOUNT,' IS ',
        CNUMMCC,'.'
* * * * * * * * *   END TEST PRINTOUT
************************************************************************
*               BEGIN ACTUAL MATCHING PROCESS FOR MOVERS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                     *                       *                       *
*                     *                       *                       *
*   *                *   *   *                *   *   *                *   *
*  *                  *   *   *                *   *   *                *  *
* *                    *  *  *                  *  *  *                  * *
**                      ***                      ***                      **
*                        *                        *                        *
*  FIRST, GENERATE SOME VARIABLES TO BE USED IN THE MATCHING PROCESS.
       DO 20 J = 1,NUMFIT
          MOS = PMOS(J)
          LAST1M = MOS(4:4)
          EXPEXC = LEXP(J) // HEXP(J)
*        WRITE(02,*) 'J = ',J,'     L = ',L
* NOW COMES THE CATEGORIZATION OF EACH E1E2 CARD AS IT IS MATCHED TO THE
* DEMAND.  EACH CARD IS FIRST SEPARATED ACCORDING TO THE NATURE OF THE
* PMOS. THERE ARE THREE MAJOR MOS CATEGORIES: PURELY NUMERIC, PARTIALLY
* OR COMPLETELY CHARACTER (THAT IS, WITH SOME OR ALL "*"'S), AND
* MISSING (THAT IS, NOTHING IN THE PMOS POSITION - MATCHES ARE MADE ON
* THE OFFTYP RATHER THAN THE PMOS.)
************   CHECK IF A NON-MOS CARD (IE. MISSING PMOS)   ************
       IF (PMOS(J) .EQ. '   0') THEN
* NOTE:THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO NON-MOS FILE
*      WRITE(14,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
***************   CHECK IF A CHARACTER MOS CARD   ***************
       ELSE IF (LAST1M .EQ. '*') THEN
* NOTE: THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO HARD-MOS FILE
*      WRITE(04,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
***************   CHECK IF PURELY NUMERIC MOS CARD   ***************
       ELSE IF ((PMOS(J) .NE. '   0') .AND. (LAST1M .NE. '*')) THEN
* NOTE: THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO EASY FILE
*      WRITE(03,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
*    NOW THE E1 CARDS HAVE BEEN SEPARATED INTO THOSE WITH PURELY NUMERIC
* MOS'S, THOSE WHICH DIFFERENTIATE BY OCC FIELD OR AMOS, AND THOSE WITH
* ONLY AN OFFTYP.  IN THIS SEGMENT OF THE PROGRAM, THE PURELY NUMERIC
* MOS'S ARE MATCHED.
*     THE ONLY TIME THE EXPERIENCE CODES WILL COME INTO PLAY IS WHEN
* THERE IS A DEMAND FOR EXPERIENCE IN THE LOWER GRADE. THIS APPEARS AS
* AN EXPERIENCE EXCEPTION CODE (EXPEXC) OF 'E ' OR 'EE'.
* WE WILL FIRST EXAMINE THE OTHER, MORE COMMON CASES
*****   LOCALIZE THE EXACT CATEGORY WITHIN THE NUMERIC MOS CARDS
*****   AND WRITE ALL MATCHES TO A FILE
************************************************************************
          IF ((AMOS1(J) .EQ. '****') .AND. (AMOS2(J) .EQ. '****')) THEN
            IF ((EXPEXC .NE. 'E ') .AND. (EXPEXC .NE. 'EE')) THEN
              IF (LDO(J) .EQ. 0) THEN
                IF (SEX(J) .EQ. 0) THEN
                  DO 30 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C            HGRD(J))
                     IF (L .EQ. 0) GOTO 35
                     IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                       LSTBOD(L) = BODNUM(J)
                       DO 35 K = 1,NUMMCC
                         ICOST = COST(ICOSTC(L),CSTCTR(K))
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
1004  FORMAT(I5,1X,A3,1X,A3,1X,I2,1X,I2,1X,I3,1X,I5,1X,I4)
35                    CONTINUE
                     ENDIF
30               CONTINUE
                ELSE IF (SEX(J) .EQ. 1) THEN
                  DO 40 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C            HGRD(J))
```

```
                  IF (L .EQ. 0) GOTO 45
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF (ISEX(L) .EQ. 1) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 45 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 45               CONTINUE
                    ENDIF
                  ENDIF
 40          CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 50 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 55
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF (ISEX(L) .EQ. 2) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 55 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 55               CONTINUE
                    ENDIF
                  ENDIF
 50          CONTINUE
              ENDIF

            ELSE IF (LDO(J) .EQ. 1) THEN
              IF (SEX(J) .EQ. 0) THEN
                DO 60 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 65
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF (ILDO(L) .EQ. 1) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 65 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 65               CONTINUE
                    ENDIF
                  ENDIF
 60          CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 70 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 75
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ.1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 75 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 75               CONTINUE
                    ENDIF
                  ENDIF
 70          CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 80 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 85
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ.1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 85 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                     FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 85                      CONTINUE
                           ENDIF
                         ENDIF
 80                CONTINUE
                    ENDIF
                 ELSE IF (LDO(J) .EQ. 2) THEN
                    IF (SEX(J) .EQ. 0) THEN
                       DO 90 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C                 HGRD(J))
                         IF (L .EQ. 0) GOTO 95
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF (ILDO(L) .EQ. 2) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 95 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 95                      CONTINUE
                           ENDIF
                         ENDIF
 90                CONTINUE
                    ELSE IF (SEX(J) .EQ. 1) THEN
                       DO 100 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C                 HGRD(J))
                         IF (L .EQ. 0) GOTO 105
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ. 2)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 105 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 105                      CONTINUE
                           ENDIF
                         ENDIF
 100               CONTINUE
                    ELSE IF (SEX(J) .EQ. 2) THEN
                       DO 110 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C                 HGRD(J))
                         IF (L .EQ. 0) GOTO 115
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ. 2)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 115 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 115                     CONTINUE
                           ENDIF
                         ENDIF
 110               CONTINUE
                    ENDIF
                 ENDIF
***
               ELSE IF ((EXPEXC .EQ. 'E ') .OR. (EXPEXC .EQ. 'EE')) THEN
                 IF (LDO(J) .EQ. 0) THEN
                    IF (SEX(J) .EQ. 0) THEN
                       DO 120 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C                 HGRD(J))
                         IF (L .EQ. 0) GOTO 125
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF (IEXP(L) .EQ. 1) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 125 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 125                     CONTINUE
                           ENDIF
```

```
                        ENDIF
 120                CONTINUE
                     ELSE IF (SEX(J) .EQ. 1) THEN
                       DO 130 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
      C               HGRD(J))
                         IF (L .EQ. 0) GOTO 135
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ISEX(L) .EQ. 1) .AND. (IEXP(L) .EQ. 1)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 135 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
      C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 135                     CONTINUE
                           ENDIF
                         ENDIF
 130                CONTINUE
                     ELSE IF (SEX(J) .EQ. 2) THEN
                       DO 140 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
      C               HGRD(J))
                         IF (L .EQ. 0) GOTO 145
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ISEX(L) .EQ. 2) .AND. (IEXP(L) .EQ. 1)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 145 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
      C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 145                     CONTINUE
                           ENDIF
                         ENDIF
 140                CONTINUE
                     ENDIF
                  ELSE IF (LDO(J) .EQ. 1) THEN
                     IF (SEX(J) .EQ. 0) THEN
                       DO 150 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
      C               HGRD(J))
                         IF (L .EQ. 0) GOTO 155
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ILDO(L) .EQ. 1) .AND. (IEXP(L) .EQ. 1)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 155 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
      C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 155                     CONTINUE
                           ENDIF
                         ENDIF
 150                CONTINUE
                     ELSE IF (SEX(J) .EQ. 1) THEN
                       DO 160 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
      C               HGRD(J))
                         IF (L .EQ. 0) GOTO 165
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                           IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ.1)
      C                  .AND. (IEXP(L) .EQ. 1)) THEN
                             LSTBOD(L) = BODNUM(J)
                             DO 165 K = 1,NUMMCC
                               ICOST = COST(ICOSTC(L),CSTCTR(K))
                               WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
      C                       FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 165                     CONTINUE
                           ENDIF
                         ENDIF
 160                CONTINUE
                     ELSE IF (SEX(J) .EQ. 2) THEN
                       DO 170 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
      C               HGRD(J))
                         IF (L .EQ. 0) GOTO 175
                         IF (LSTBOD(L) .NE. BODNUM(J)) THEN
```

```
                    IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ.1)
     C            .AND. (IEXP(L) .EQ. 1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 175 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 175              CONTINUE
                    ENDIF
                  ENDIF
 170          CONTINUE
              ENDIF
            ELSE IF (LDO(J) .EQ. 2) THEN
              IF (SEX(J) .EQ. 0) THEN
                DO 180 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 185
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ILDO(L) .EQ. 2) .AND. (IEXP(L) .EQ. 1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 185 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 185              CONTINUE
                    ENDIF
                  ENDIF
 180          CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 190 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 195
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ. 2)
     C            .AND. (IEXP(L) .EQ. 1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 195 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 195              CONTINUE
                    ENDIF
                  ENDIF
 190          CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 200 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 205
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ. 2)
     C            .AND. (IEXP(L) .EQ. 1)) THEN
                      LSTBOD(L) = BODNUM(J)
                      DO 205 K = 1,NUMMCC
                        ICOST = COST(ICOSTC(L),CSTCTR(K))
                        WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
 205              CONTINUE
                    ENDIF
                  ENDIF
 200          CONTINUE
              ENDIF
            ENDIF
          ENDIF
********
* IN HERE WE WILL LATER INSERT A SERIES OF "IF-THEN-ELSE-IF"
* STATEMENTS TO COVER ALL CONDITIONS UNDER THE MAJOR CONDITION:
*        IF ((AMOS1(J) .EQ. '****') .AND. (AMOS2(J) .EQ. '****')) THEN
* THIS WILL CAPTURE THOSE CARDS WHICH HAVE A PARTIALLY NUMERIC OR
* COMPLETELY CHARACTER (IE.:'****') AMOS.  FOR NOW, WE WILL IGNORE THEM.
```

```
          ENDIF
        ENDIF

*     NOW THAT WE HAVE MATCHED ALL ACCEPTABLE OFFICERS FROM THE
* INVENTORY TO CARD(J), WE GO BACK TO THE BEGINNING OF THE MATCHING
* PROCESS UNTIL WE HAVE LOOKED AT ALL FIT LEVELS FOR THE GROUP OF
* DEMANDS WE HAVE READ IN.
20    CONTINUE
*  WE ARE ABOUT TO GO THE NEXT GROUP IN THE FILE. BUT SINCE WE FOUND THE
* END OF THE LAST GROUP WHEN WE HIT THE FIRST LINE IN THE NEXT GROUP,
* WE MUST RECOVER THE VALUES OF THE FIRST CARD IN THE NEW GROUP WHICH
* WERE STORED IN TEMPORARY VARIABLES.
        I = 1
        LGRD(I) = TLGRD
        LEXP(I) = TLEXP
        HGRD(I) = THGRD
        HEXP(I) = THEXP
        PMOS(I) = TPMOS
        OFFTYP(I) = TOFTYP
        AMOS1(I) = TAMOS1
        AMOS2(I) = TAMOS2
        LDO(I) = TLDO
        SEX(I) = TSEX
        MOSNUM(I) = TMOSNO
        E1NUM(I) = TE1NUM
        BODNUM(I) = TBODA
        CSTCTR(I) = TCCTR
        E2NUM(I) = TE2NUM
        FMCC(I) = TFMCC
        SPL(I) = TSPL
        FITLVL(I) = TFITLV
        CAPCTY(I) = TCAP
        TKOUNT = TKOUNT + 1
        GOTO 501
9993  WRITE(02,*) 'GROUP ',TKOUNT,' HAS ',ENDGRP,' CARDS.'
********************   BEGIN MATCH OF NON-MOVERS   ********************
        K = 1
**********   READ NON-MOVER INVENTORY INTO RESIDENT MEMORY   ***********
        DO 7 I = 1,20000
          READ(11,102,END=9994) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
     CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
        LSTBOD(I) = 0
        IF (K .EQ. 1000) THEN
         WRITE(02,*) 'K =',K
         WRITE(02,102) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
     CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
        ELSE IF (K .EQ. 2000) THEN
         WRITE(02,*) 'K =',K
         WRITE(02,102) IMOS(I),IGRD(I),IAMOS1(I),IAMOS2(I),
     CPMCC(I),IEXP(I),ISEX(I),ILDO(I),IDNUM(I),IMOSNO(I),ICOSTC(I)
        ENDIF
        K=K+1
7     CONTINUE
9994  CONTINUE
        WRITE(02,*) 'THERE ARE',K,'PEOPLE IN THE NON-MOVER INVENTORY FILE''
        SUPSIZ = SUPSIZ + K
***  OUTPUT THE NUMBER OF PEOPLE IN THE INVENTORY (= SUPPLY SIZE)    ***
***  TO A DATA FILE.  THIS WILL BE USED IN THE ARC GENERATOR PROGRAM ***
        WRITE (17,104) SUPSIZ
104   FORMAT(I5)
***   WHILE WE ARE THINKING OF THE ARC GENERATOR ROUTINE, WE MIGHT   ***
***   AS WELL ALSO OUTPUT THE OTHER INPUT ITEM FOR THAT PROGRAM.     ***
***   WE WANT TO ALLOW THE DECISION MAKER TO ADJUST THE WEIGHTS OF   ***
***   THE FIT AND COST OBJECTIVES IN THE NETWORK FORMULATION.  FOR   ***
***   THIS, WE WILL NEED AN ALPHA VALUE WHICH CAN BE ADJUSTED. AT    ***
***   FIRST, WE WILL SET ALPHA TO 1.0 WHICH CORRESPONDS TO MAKING     ***
***   THE FIT OBJECTIVE PRE-EMPTIVE OVER THE COST.  THIS VALUE WILL   ***
***   BE READ OUT OF THE FILE ALPHA1 DATA A1 WHEN THE PROGRAM BEGINS.***
***   ONCE THE PROBLEM HAS BEEN RUN ONCE, FUTURE VALUES OF ALPHA      ***
***   BE READ OUT OF A DIFFERENT FILE, ALPHA2 DATA A1, WHICH          ***
***   CONTAINS EITHER THE ORIGINAL ALPHA VALUE (IF NO CHANGE IN THE   ***
***   WEIGHTS HAS BEEN MADE) OR A VALUE ADJUSTED BY THE DECISION      ***
***   TO REFLECT A DIFFERENT WEIGHT ON THE OBJECTIVES.   SO... NOW    ***
***   WE WILL OUTPUT A VALUE OF ALPHA = 1.0                           ***
       ALPHA = 1
       WRITE(18,106) ALPHA
106   FORMAT(F5.3)
********************   RE-INITIALIZE EP ARRAY   *******************
       DO 22 I = 1,250
         DO 22 J = 1,5
           BEGSCH(I,J) = 0
           ENDSCH(I,J) = 0
22    CONTINUE
**************   READ EP ARRAY INTO RESIDENT MEMORY   ****************
*     NEXT, WE WILL READ IN THE ENTRY POINT ARRAY DEVELOPED FOR THE
* NON-MOVERS, CALLED NMOVR-EP ARRAY
       DO 8 I = 1,250
         DO 8 J = 1,5
           READ(19,103,END=9995) BEGSCH(I,J),ENDSCH(I,J)
* * * * * * * *   BEGIN TEST PRINTOUT
       WRITE(02,*) I,J,'BEGSCH(I,J)=',BEGSCH(I,J),'ENDSCH(I,J)=',
      CENDSCH(I,J)
* * * * * * * *   END TEST PRINTOUT
8     CONTINUE
9995  CONTINUE
**************************************************************************
*     READ IN EACH LINE FROM THE FILE MOVR-DEM INPUT FOR MATCHING        *
*       READ IN E1/E2/ASR CARDS AND SORT OUT THE NUMBERS OF FIT LEVELS
*       AND MCC'S WITH EACH PARTICULAR BOD DEMAND.
*     INITIALIZE
       I = 1
       TOTFIT = 0
       READ(13,101) LGRD(I),LEXP(I),HGRD(I),HEXP(I),PMOS(I),OFFTYP(I),
      CAMOS1(I),AMOS2(I),LDO(I),SEX(I),MOSNUM(I),E1NUM(I),BODNUM(I),
      CE2NUM(I),CSTCTR(I),FMCC(I),SPL(I),FITLVL(I),CAPCTY(I)
       TKOUNT = 1
601   I = I+1
       READ(13,101,END=9996) LGRD(I),LEXP(I),HGRD(I),HEXP(I),PMOS(I),
      COFFTYP(I),AMOS1(I),AMOS2(I),LDO(I),SEX(I),MOSNUM(I),E1NUM(I),
      CBODNUM(I),E2NUM(I),CSTCTR(I),FMCC(I),SPL(I),FITLVL(I),CAPCTY(I)
       IF (BODNUM(I) .EQ. BODNUM(I-1)) GOTO 601
       IF (BODNUM(I) .NE. BODNUM(I-1)) THEN
         ENDGRP = I-1
*     STORE THE FIRST VARIABLE IN THE NEXT BOD SET IN TEMPORARY VALUES
           TLGRD = LGRD(I)
           TLEXP = LEXP(I)
           THGRD = HGRD(I)
           THEXP = HEXP(I)
           TPMOS = PMOS(I)
           TOFTYP = OFFTYP(I)
           TAMOS1 = AMOS1(I)
           TAMOS2 = AMOS2(I)
           TLDO = LDO(I)
           TSEX = SEX(I)
           TMOSNO = MOSNUM(I)
           TE1NUM =  E1NUM(I)
           TBODA = BODNUM(I)
           TCCTR = CSTCTR(I)
           TE2NUM = E2NUM(I)
           TFMCC = FMCC(I)
           TSPL = SPL(I)
```

```
            TFITLV = FITLVL(I)
            TCAP = CAPCTY(I)
         ENDIF
* * * * * * * *   BEGIN TEST PRINTOUT
         WRITE(02,*) '*************************************************'
         WRITE(02,*) 'GROUP ',TKOUNT,' HAS ',ENDGRP,' CARDS.'
* * * * * * * *   END TEST PRINTOUT
*     EACH OF THE GROUPS WE HAVE JUST READ IN CONTAIN ALL OF THE E1-CARD
*   AND ASR INFORMATION FOR THE MOVERS.  EACH GROUP CONTAINS ALL OF THE
*   E1-CARD DATA AND THE MCC AND DEMAND (OR CAPACITY) FOR EACH DEMAND
*   ASSOCIATED WITH THAT E1 INFORMATION.  BECAUSE WE HAVE COMPRESSED SO
*   MUCH INFORMATION INTO A SINGLE FILE, WE MUST NOW DETERMINE TWO THINGS
*   ABOUT EACH GROUP.  FIRST, WE MUST FIND THE NUMBER OF SUBSTITUTION
*   LEVELS FOR THE DEMANDS OF THIS TYPE (ALSO CALLED THE "BOD SET").
*   ADDITIONALLY, WE MUST ASCERTAIN THE NUMBER OF DEMANDS IN THE GROUP
*   WHICH CORRESPOND TO THE BOD SET.  WE WILL USE THESE TWO ITEMS OF
*   INFORMATION AS INDICES TO HELP US PERFORM THE MATCHING OF PEOPLE TO
*   BILLETS WITH SOME SEMBLANCE OF EFFICIENCY. THE NUMBER OF SUBSTITUTION
*   LEVELS, CALLED "NUMFIT", TELLS US WHEN TO BEGIN SEARCHING THROUGH THE
*   NEXT BOD.  THE NUMBER OF DEMAND LOCATIONS, WHICH IS EQUIVALENT TO THE
*   NUMBER OF MCC'S, IS CALLED "NUMMCC".  THIS INDEX TELLS US HOW MANY
*   (AND WHICH) MCC'S WILL BE MATCHED TO THE VALID PEOPLE-SUBSTITUTION
*   MATCHES WE FIND BY MATCHING THE BOD SETS TO THE INVENTORY.
         MARKR1 = 0
         MARKR2 = 0
         IF (ENDGRP .EQ. 1) THEN
           NUMFIT = 1
           NUMMCC = 1
         ELSE IF (ENDGRP .GT. 1) THEN
           DO 210 I = 2,ENDGRP
             IF ((FITLVL(I) .EQ. FITLVL(I-1)) .AND. (MARKR1 .EQ. 0)) THEN
               NUMFIT = I-1
               MARKR1 = 1
             ELSE IF ((FITLVL(I) .GT. FITLVL(I-1)) .AND. (I .EQ. ENDGRP))
        CTHEN
               NUMFIT = I
             ENDIF
             IF ((FMCC(I) .EQ. FMCC(I-1)) .AND. (MARKR2 .EQ. 0)) THEN
               NUMMCC = I-1
               MARKR2 = 1
             ELSE IF ((FMCC(I) .NE. FMCC(I-1)) .AND. (I .EQ. ENDGRP)) THEN
               NUMMCC = I
             ENDIF
210     CONTINUE
           ENDIF
*      AT THIS POINT, WE SHOULD HAVE THE NUMBER OF FIT LEVELS AND DEMANDS
*    WITHIN THE GROUP. AS A CHECK, WE WILL PRINT OUT THE RESULTS.
*    WE WILL INCLUDE A RUNNING TOTAL OF THE TOTAL NUMBER OF E1-CARDS
*    INCLUDED IN THE FILE, AND CALL THIS VARIABLE TOTFIT.
         TOTFIT = TOTFIT + NUMFIT
* * * * * * * *   BEGIN TEST PRINTOUT
         WRITE(02,*) 'THE NUMBER OF FIT LEVELS IN GROUP ',TKOUNT,' IS ',
        CNUMFIT,'.'
         WRITE(02,*) 'SO FAR THERE HAVE BEEN A TOTAL OF',TOTFIT,'E1 CARDS.'
         WRITE(02,*) 'THE NUMBER OF MCC DEMANDS IN GROUP ',TKOUNT,' IS ',
        CNUMMCC,'.'
* * * * * * * *   END TEST PRINTOUT
************************************************************************
*                               MATCH NON-MOVERS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                      *                        *                      *
*                      *                        *                      *
*  *                *  *  *                  *  *  *                *  *
* *                  * * *                    * * *                  * *
* *                   * * *                    * * *                  * *
**                     ***                      ***                   **
```

```
*                   *                    *                    *
*  NOW GENERATE SOME VARIABLES TO BE USED IN THE MATCHING PROCESS.
      DO 220 J = 1,NUMFIT
         MOS = PMOS(J)
         LAST1M = MOS(4:4)
         EXPEXC = LEXP(J) // HEXP(J)
*       WRITE(02,*) 'LAST1M =',LAST1M,'   EXPEXC = ',EXPEXC
* NOW COMES THE CATEGORIZATION OF EACH E1E2 CARD AS IT IS MATCHED TO THE
* DEMAND.  EACH CARD IS FIRST SEPARATED ACCORDING TO THE NATURE OF THE
* PMOS. THERE ARE THREE MAJOR MOS CATEGORIES: PURELY NUMERIC, PARTIALLY
* OR COMPLETELY CHARACTER (THAT IS, WITH SOME OR ALL "*"'S), AND
* MISSING (THAT IS, NOTHING IN THE PMOS POSITION - MATCHES ARE MADE ON
* THE OFFTYP RATHER THAN THE PMOS.)
************    CHECK IF A NON-MOS CARD (IE. MISSING PMOS)    ************
      IF (PMOS(J) .EQ. '    0') THEN
* NOTE:THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO NON-MOS FILE
*       WRITE(07,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
***************    CHECK IF A CHARACTER MOS CARD    ***************
      ELSE IF (LAST1M .EQ. '*') THEN
* NOTE: THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO HARD-MOS FILE
*       WRITE(04,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
***************    CHECK IF PURELY NUMERIC MOS CARD    ***************
      ELSE IF  (PMOS(J) .NE.    0') .AND. (LAST1M .NE. '*')) THEN
* NOTE: THE NEXT THREE LINES MAY BE REMOVED - OUTPUT TO EASY FILE
*       WRITE(03,101) LGRD(J),LEXP(J),HGRD(J),HEXP(J),PMOS(J),OFFTYP(J),
*    CAMOS1(J),AMOS2(J),LDO(J),SEX(J),MOSNUM(J),E1NUM(J),BODNUM(J),
*    CBODE2(J),E2NUM(J),FMCC(J),SPL(J),FITLVL(J),CAPCTY(J)
*    WE NOW HAVE SEPARATED THOSE E1 CARDS WHICH HAVE PURELY NUMERIC FROM
* THOSE WITH ASTERISKS IN ONE OR MORE OF THE MOS FIELDS, AND THOSE WITH
* ONLY AN OFFTYP.  IN THIS SEGMENT OF THE PROGRAM, WE ARE MATCHING THE
* PURELY NUMERIC MOS E1 CARDS.
*      THE ONLY TIME THE EXPERIENCE CODES WILL COME INTO PLAY IS WHEN
* THERE IS A DEMAND FOR EXPERIENCE IN THE LOWER GRADE. THIS APPEARS AS
* AN EXPERIENCE EXCEPTION CODE (EXPEXC) OF 'E ' OR 'EE'.
* WE WILL FIRST EXAMINE THE OTHER, MORE COMMON CASES
*    SINCE ONLY NON-MOVERS ARE BEING CONSIDERED IN THIS PART OF THE
* PROGRAM, THE MOVING COST ("ICOST") IS ALWAYS ZERO.
      ICOST = 0
*****    LOCALIZE THE EXACT CATEGORY WITHIN THE NUMERIC MOS CARDS
**********************************************************************
          IF ((AMOS1(J) .EQ. '****') .AND. (AMOS2(J) .EQ. '****')) THEN
            IF  (EXPEXC .NE. 'E ') .AND. (EXPEXC .NE. 'EE')) THEN
              IF (LDO(J) .EQ. 0) THEN
                IF (SEX(J) .EQ. 0) THEN
                  DO 230 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C            HGRD(J))
                    IF (L .EQ. 0) GOTO 235
                    DO 235 K = 1,NUMMCC
                      IF (PMCC(L) .EQ. FMCC(K)) THEN
                        IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                          LSTBOD(L) = BODNUM(J)
                            WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K)
     C                     ,FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
235               CONTINUE
230             CONTINUE
                ELSE IF (SEX(J) .EQ. 1) THEN
                  DO 240 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C            HGRD(J))
                    IF (L .EQ. 0) GOTO 245
                    DO 245 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF (ISEX(L) .EQ. 1) THEN
                         LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
245            CONTINUE
240          CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 250 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 255
                    DO 255 K = 1,NUMMCC
                      IF (PMCC(L) .EQ. FMCC(K)) THEN
                       IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                         IF (ISEX(L) .EQ. 2) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                         ENDIF
                       ENDIF
                      ENDIF
255              CONTINUE
250          CONTINUE
              ENDIF
            ELSE IF (LDO(J) .EQ. 1) THEN
              IF (SEX(J) .EQ. 0) THEN
                DO 260 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 265
                  DO 265 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF (ILDO(L) .EQ. 1) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
265            CONTINUE
260          CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 270 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 275
                  DO 275 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ.1)) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
275            CONTINUE
270          CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 280 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 285
                  DO 285 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ.1)) THEN
                    LSTBOD(L) = BODNUM(J)
                    WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C            FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                  ENDIF
                ENDIF
              ENDIF
 285        CONTINUE
 280      CONTINUE
          ENDIF
        ELSE IF (LDO(J) .EQ. 2) THEN
          IF (SEX(J) .EQ. 0) THEN
            DO 290 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C      HGRD(J))
              IF (L .EQ. 0) GOTO 295
              DO 295 K = 1,NUMMCC
                IF (PMCC(L) .EQ. FMCC(K)) THEN
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF (ILDO(L) .EQ. 2) THEN
                      LSTBOD(L) = BODNUM(J)
                      WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C              FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                    ENDIF
                  ENDIF
                ENDIF
 295          CONTINUE
 290        CONTINUE
          ELSE IF (SEX(J) .EQ. 1) THEN
            DO 300 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C      HGRD(J))
              IF (L .EQ. 0) GOTO 305
              DO 305 K = 1,NUMMCC
                IF (PMCC(L) .EQ. FMCC(K)) THEN
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ. 2)) THEN
                      LSTBOD(L) = BODNUM(J)
                      WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C              FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                    ENDIF
                  ENDIF
                ENDIF
 305          CONTINUE
 300        CONTINUE
          ELSE IF (SEX(J) .EQ. 2) THEN
            DO 310 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C      HGRD(J))
              IF (L .EQ. 0) GOTO 315
              DO 315 K = 1,NUMMCC
                IF (PMCC(L) .EQ. FMCC(K)) THEN
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ. 2)) THEN
                      LSTBOD(L) = BODNUM(J)
                      WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C              FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                    ENDIF
                  ENDIF
                ENDIF
 315          CONTINUE
 310        CONTINUE
          ENDIF
        ENDIF
      ELSE IF ((EXPEXC .EQ. 'E ') .OR. (EXPEXC .EQ. 'EE')) THEN
        IF (LDO(J) .EQ. 0) THEN
          IF (SEX(J) .EQ. 0) THEN
            DO 320 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C      HGRD(J))
              IF (L .EQ. 0) GOTO 325
              DO 325 K = 1,NUMMCC
                IF (PMCC(L) .EQ. FMCC(K)) THEN
                  IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                    IF (IEXP(L) .EQ. 1) THEN
```

```
                        LSTBOD(L) = BODNUM(J)
                       WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C               FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                      ENDIF
                    ENDIF
                  ENDIF
 325           CONTINUE
 320         CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 330 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 335
                  DO 335 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                       IF ((ISEX(L) .EQ. 1) .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                       ENDIF
                      ENDIF
                    ENDIF
 335           CONTINUE
 330         CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 340 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 345
                  DO 345 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                       IF ((ISEX(L) .EQ. 2) .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                       ENDIF
                      ENDIF
                    ENDIF
 345           CONTINUE
 340         CONTINUE
              ENDIF
            ELSE IF (LDO(J) .EQ. 1) THEN
              IF (SEX(J) .EQ. 0) THEN
                DO 350 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 355
                  DO 355 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                       IF ((ILDO(L) .EQ. 1) .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                       ENDIF
                      ENDIF
                    ENDIF
 355           CONTINUE
 350         CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 360 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 365
                  DO 365 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                       IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ.1)
     C                 .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                         WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                 FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
```

```
                      ENDIF
                    ENDIF
                  ENDIF
365             CONTINUE
360           CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 370 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 375
                  DO 375 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ.1)
     C                  .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                          WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                    FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
375             CONTINUE
370           CONTINUE
              ENDIF
            ELSE IF (LDO(J) .EQ. 2) THEN
              IF (SEX(J) .EQ. 0) THEN
                DO 380 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 385
                  DO 385 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ILDO(L) .EQ. 2) .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                          WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                    FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
385             CONTINUE
380           CONTINUE
              ELSE IF (SEX(J) .EQ. 1) THEN
                DO 390 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 395
                  DO 395 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ISEX(L) .EQ. 1) .AND. (ILDO(L) .EQ. 2)
     C                  .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                          WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                    FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
                      ENDIF
                    ENDIF
395             CONTINUE
390           CONTINUE
              ELSE IF (SEX(J) .EQ. 2) THEN
                DO 400 L = BEGSCH(MOSNUM(J),LGRD(J)),ENDSCH(MOSNUM(J),
     C          HGRD(J))
                  IF (L .EQ. 0) GOTO 405
                  DO 405 K = 1,NUMMCC
                    IF (PMCC(L) .EQ. FMCC(K)) THEN
                      IF (LSTBOD(L) .NE. BODNUM(J)) THEN
                        IF ((ISEX(L) .EQ. 2) .AND. (ILDO(L) .EQ. 2)
     C                  .AND. (IEXP(L) .EQ. 1)) THEN
                          LSTBOD(L) = BODNUM(J)
                          WRITE(10,1004) IDNUM(L),PMCC(L),FMCC(K),SPL(K),
     C                    FITLVL(J),CAPCTY(K),ICOST,E2NUM(K)
                        ENDIF
```

```
                          ENDIF
                        ENDIF
405                   CONTINUE
400                 CONTINUE
                     ENDIF
                   ENDIF
                 ENDIF
********
* IN HERE WE WILL LATER INSERT A SERIES OF "IF-THEN-ELSE-IF"
* STATEMENTS TO COVER ALL CONDITIONS UNDER THE MAJOR CONDITION:
*       IF ((AMOS1(J) .EQ. '****') .AND. (AMOS2(J) .EQ. '****')) THEN
* THIS WILL CAPTURE THOSE CARDS WHICH HAVE A PARTIALLY NUMERIC OR
* COMPLETELY CHARACTER (IE.:'****') AMOS.  FOR NOW, WE WILL IGNORE THEM.
          ENDIF
        ENDIF

*     NOW THAT WE HAVE MATCHED ALL ACCEPTABLE OFFICERS FROM THE
* INVENTORY TO CARD(J), WE GO BACK TO THE BEGINNING OF THE MATCHING
* PROCESS UNTIL WE HAVE LOOKED AT ALL FIT LEVELS FOR THE GROUP OF
* DEMANDS WE HAVE READ IN.
220   CONTINUE
*  WE ARE ABOUT TO GO THE NEXT GROUP IN THE FILE. BUT SINCE WE FOUND THE
* END OF THE LAST GROUP WHEN WE HIT THE FIRST LINE IN THE NEXT GROUP,
* WE MUST RECOVER THE VALUES OF THE FIRST CARD IN THE NEW GROUP WHICH
* WERE STORED IN TEMPORARY VARIABLES.
        I = 1
        LGRD(I) = TLGRD
        LEXP(I) = TLEXP
        HGRD(I) = THGRD
        HEXP(I) = THEXP
        PMOS(I) = TPMOS
        OFFTYP(I) = TOFTYP
        AMOS1(I) = TAMOS1
        AMOS2(I) = TAMOS2
        LDO(I) = TLDO
        SEX(I) = TSEX
        MOSNUM(I) = TMOSNO
        E1NUM(I) = TE1NUM
        BODNUM(I) = TBODA
        CSTCTR(I) = TCCTR
        E2NUM(I) = TE2NUM
        FMCC(I) = TFMCC
        SPL(I) = TSPL
        FITLVL(I) = TFITLV
        CAPCTY(I) = TCAP
        TKOUNT = TKOUNT + 1
        GOTO 601
9996  WRITE(02,*) 'GROUP ',TKOUNT,' HAS ',ENDGRP,' CARDS.'
*     NOW WE HAVE FINISHED DEFINING THE ARCS FOR MOVERS.
* IN ORDER TO SAVE SPACE, WE WILL TRY TO OVER-WRITE THE ARRAYS WE JUST
* USED FOR MOVERS IN THE DETERMINATION OF MOVER ARCS.  BECAUSE THIS IS
* FRAUGHT WITH DANGER OF ABENDING, WE MUST BE VERY CAREFUL TO KEEP COUNT
* OF EXACTLY HOW MANY PEOPLE AND E1/E2/ASR CARDS ARE ASSOCIATED WITH THE
* NON-MOVERS.
*     WITH THAT CAVEAT IN MIND, PROCEED TO MATCH THE FIXED PART
* OF THE INVENTORY IN MUCH THE SAME WAY AS THE FREE PART.  FIRST,
* READ THE NON-MOVERS INTO THE INVENTORY ARRAY.  NEXT, READ THE
* ENTRY POINT ARRAY FOR THE NON-MOVERS INTO THE EP-ARRAY.  THEN,
* BEGIN TO READ IN THE E1/E2/ASR CARDS UNTIL WE HAVE READ IN ALL OF A
* PARTICULAR GROUP.  A GROUP IS DEFINED AS A SET CONTAINING ALL OF THE
* SUBSTITUTION CARDS (IE. FIT LEVELS) FOR A PARTICULAR BOD (BILLET) AND
* ALL OF THE DEMAND (THE ACTUAL MCC'S AND UNFILLED CAPACITIES) FOR THAT
* BOD.  WORKING WITHIN EACH GROUP, FIND EVERYONE WHO MATCHES A GIVEN
* SUBSTITUTION CRITERIA WHO HAS NOT PREVIOUSLY BEEN MATCHED WITHIN THAT
* BOD SET.  THEN GENERATE ARCS BETWEEN THAT INDIVIDUAL AND ALL MCC'S
* IN THE GROUP.  THE PERSON IS THEN MARKED SO HE WON'T BE MATCHED AGAIN
* WITHIN THE SAME BOD, AND THE SEARCH CONTINUES.  ONCE ALL INDIVIDUALS
* HAVE BEEN CHECKED FOR THAT CRITERIA, THE NEXT CARD IS READ, AND THE
* PROCESS CONTINUES.  ONCE ALL SUBSTITUTION LEVELS IN A GIVEN GROUP
* HAVE BEEN MATCHED, THE NEXT GROUP OF E1/E2/ASR CARDS IS READ.
* THE MCC'S REPRESENTED IN THE GROUP OF CARDS.
      STOP
      END
```

APPENDIX Q
BIGSORT SAS

1. PROGRAM TO SORT THE PEOPLE-JOB MATCHES OUTPUT FROM MATCH-AL FORTRAN

```
*********************************************************************
*                                                                   *
*        *  *  *  *   PROGRAM NAME: BIGSORT SAS       *  *  *  *    *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                   *
*                  * * *   OVERVIEW AND PURPOSE  * * *              *
*                                                                   *
*    THIS PROGRAM SORTS THE MATCHES OF PEOPLE-TO-BILLETS WHICH ARE  *
* OUTPUT FROM MATCH-AL FORTRAN, IN PREPARATION FOR INPUT INTO THE   *
* NETWORK GENERATION ROUTINE.  THIS SORTING FUNCTION WAS HANDLED IN *
* SAS BECAUSE OF THE CONVENIENCE OF THE SAS SORTING ROUTINES. HOWEVER, *
* THE SORT OF APPROXIMATELY 85,000 ARCS TOOK OVER 100 SECONDS OF CPU *
* TIME.  CONVERTING THE SORTING FUNCTION TO FORTRAN MIGHT RESULT IN A *
* SIGNIFICANT REDUCTION IN THAT TIME.                               *
*    THE MATCHES ARE SORTED BY SPL, E2NUM, AND FMCC, PRIMARILY.  THE *
* SPL DEFINES WHAT ARE CALLED "QUOTA GROUPS" IN THE NETWORK.  QUOTA *
* GROUPS ARE ALL BILLETS WITH THE SAME SPL.  THE E2NUM (WHICH IS    *
* EQUIVALENT TO THE BOD) AND THE FMCC DEFINE THE SPECIFIC DEMANDS   *
* FOR THE NETWORK.  THESE DEMANDS ARE CALLED "QUOTAS" IN THE NETWORK. *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                   *
*                 ***   DEFINITION OF TERMS   ***                   *
*                                                                   *
*   CAPCTY    THE NUMBER OF BILLETS OF THE PARTICULAR BOD (E2NUM) AT *
*             THE LOCATION (FMCC)                                   *
*   COST      PCS COST FOR THE PERSON-TO-JOB MATCH                  *
*   E2NUM     INDEXES THE SPECIFIC JOB DESCRIPTION (BOD) FOR THE MATCH *
*   FITLVL    FIT LEVEL OF THE MATCH BASED ON SUBSTITUTION PRIORITY *
*   FMCC      FUTURE MCC.  THE LOCATION OF THE DEMAND IN THE MATCH  *
*   IDNUM     THE INDEX OF THE INDIVIDUAL ON THE INVENTORY LIST     *
*   SPL       THS STAFFING PRECEDENCE LEVEL OF THE DEMAND IN THE MATCH *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *     FILE DEFINITION     *  *  *  *  *  *  *  *  *
CMS FILEDEF FONE DISK MATCH OUTPUT A;
CMS FILEDEF FTWO DISK SORTED RAW-ARCS A (RECFM F LRECL 35 BLOCK 35;
*********************************************************************
DATA DONE;
    INFILE FONE;
    INPUT IDNUM 1-5 PMCC $7-9 FMCC $11-13 SPL 15-16 FITLVL 18-19 CAPCTY
        21-23 COST 25-29 E2NUM 31-34;
    IF IDNUM = . THEN DELETE;
    IF PMCC =  ' ' THEN DELETE;
    IF FMCC = ' ' THEN DELETE;
    IF SPL = . THEN DELETE;
    IF FITLVL = . THEN DELETE;
    IF CAPCTY = . THEN DELETE;
    IF COST = . THEN DELETE;
    IF E2NUM = . THEN DELETE;
PROC SORT DATA=DONE;
    BY SPL E2NUM FMCC FITLVL COST IDNUM;
DATA _NULL_;
    SET DONE;
    FILE FTWO;
        PUT SPL 1-2 E2NUM 4-7 FMCC $9-11 CAPCTY 13-15 FITLVL 17-18 COST
        20-24 IDNUM 26-30;
```

APPENDIX R
NET-GENX FORTRAN

1. PROGRAM TO GENERATE CAPACITATED TRANSSHIPMENT NETWORK

```
***********************************************************************
*                                                                     *
*      *  *  *  *   PROGRAM NAME: NET-GENX FORTRAN    *  *  *  *      *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                *  *  *   OVERVIEW AND PURPOSE   *  *  *             *
*                                                                     *
*    IN THIS PROGRAM THE RAW ARC INFORMATION GENERATED IN THE MATCHING *
* PROGRAM (MATCH-AL FORTRAN) AND SORTED BY SPL, E2NUM, MCC, FITLVL,   *
* AND COST IN BIGSORT SAS A1, IS PUT INTO GNET FORMAT AS A CAPACITATED *
* TRANSSHIPMENT NETWORK, WITH APPROPRIATE COSTS AND LABELS.  ONE MAJOR *
* GOAL IS TO GIVE THE DECISION MAKER THE FLEXIBILITY TO ADJUST TARGETS *
* (IN THIS CASE, THE ACCEPTABILITY BOUND ON FILL) AND MODIFY THE      *
* RELATIVE WEIGHTS PLACED ON THE FIT AND COST OBJECTIVES.  (IN EFFECT *
* THIS  SOLVES  THE PROBLEM WITH FIT AS A PRE-EMPTIVE OBJECTIVE OVER  *
* PCS COST,  WITH COST AS PRE-EMPTIVE OVER FIT, OR WITH SOME WEIGHTED *
* COMBINATION OF THE TWO).  TO DO THIS TWO CAPABILITIES ARE NEEDED.   *
* FIRST, THE DECISION MAKER MUST BE ABLE TO SET THE UPPER BOUND ON    *
* FILL TO SOME LIMITING VALUE.  SINCE THE SOLVER CANNOT FILL ANY MORE *
* THAN THE UPPER BOUND, ANY ADDITIONAL DUAL DEGENERACY CREATED BY     *
* IMPOSING A REDUCED UPPER BOUND CAN BE USED BY THE SOLVER TO IMPROVE *
* THE FIT AND COST OBJECTIVES.  WHILE LIMITED TO THE FILL OBJECTIVE   *
* IN THIS PROTOTYPE, THIS PRINCIPLE COULD BE EXPANDED TO INCLUDE MANY *
* OF THE OTHER OBJECTIVES AS WELL (IE. THOSE EXPRESSIBLE AS FLOW ON A *
* SINGLE ARC.)                                                        *
*        AN ALPHA VALUE WHICH CAN BE ADJUSTED BY THE USER WILL BE USED *
* TO CONTROL THE RELATIVE PRIORITIES OF THE FIT AND COST OBJECTIVES.  *
* INITIALLY, THE VALUE OF ALPHA IS SET TO 1.0, WHICH CORRESPONDS TO   *
* MAKING FIT A PRE-EMPTIVE OBJECTIVE OVER COST.  THIS VALUE CAN BE    *
* ADJUSTED BY THE DECISION MAKER UPON VIEWING THE INITIAL SOLUTION.   *
* FUTURE ENHANCEMENTS COULD INCLUDE PROVISION FOR INTERACTIVE SETTING *
* OF ALPHA BEFORE THE FIRST RUN.  THIS IS NOT DONE HERE FOR SEVERAL   *
* REASONS.  FIRST, IT IS NOT THE PURPOSE OF THIS THESIS TO IMPLEMENT  *
* THE INTERACTIVE FEATURES OF THE MODEL.  SECOND, THE STARTING POINT  *
* IS  ARBITRARY, AND THUS THE DEFAULT WAS SET TO THE PRESENT          *
* OBJECTIVE.  THIRD, IT IS NOT CLEAR THAT THE AVERAGE USER OF THIS    *
* MODEL WOULD UNDERSTAND THE SIGNIFICANCE OF SETTING ALPHA TO A       *
* PARTICULAR VALUE.  RATHER THAN TAKE A LONG TIME TO EXPLAIN IT IN    *
* A "USER-FRIENDLY" FASHION, IT MIGHT BE BETTER TO ADJUST IT          *
* IMPLICITLY, THEREBY CREATING AN IMPLICIT UTILITY FUNCTION FOR THE   *
* USER.                                                               *
*      THIS PROGRAM IS LARGELY SELF DOCUMENTING.  THE DETAILS OF THE   *
* IMPLEMENTATION ARE DISCUSSED AS THEY APPEAR IN THE PROGRAM.         *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*              *  *  *   FILE DEFINITIONS    *  *  *                  *
*                                                                     *
*    FILEDEF   FILE IDENTIFICATION        CONTENT/PURPOSE             *
*       1        ALPHA DATA             CONTAINS USER-SET VALUE OF ALPHA *
*       2        UPPR-BND DATA          CONTAINS ANY UPPER BOUND ON FILL *
*      30        SUP-SIZE DATA          CONTAINS TOTAL NUMBER IN INVENTORY *
*      31        SORTED RAW-ARCS        OUTPUT FROM BIGSORT - ALL MATCHES *
*      32        DEBUG ARC-FILE         MONITORS INTERNAL PROGRAM FUNCTION *
*      34        GNET INPUT             CONTAINS THE CAPACITATED TRANS- *
*                                       SHIPMENT NETWORK IN GNET FORMAT *
*      35        NET-INFO DATA          EXTRACTS INFORMATION OF THE   *
*                                              NETWORK USED IN THE SUMMARY PROGRM *
*      36          TNUMQG DATA           GIVES TOTAL NUMBER OF QUOTA GROUPS *
*                                                                          *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                          *
*                      * * *   DEFINITION OF TERMS   * * *                 *
*                                                                          *
* NAME(DIM) TYPE                  DESCRIPTION                              *
* ---------|----|----------------------------------------------------------*
* AFLAG      I*2  - FLAG WHICH INDICATES IF ALPHA HAS BEEN CHANGED;       *
*                   WHEN AFLAG = 0 THE DEFAULT VALUE OF A = 1 IS USED.    *
* ALPHA       R   - THE ALPHA USED TO WEIGHT FIT AND PCS COST IN THE      *
*                   OBJECTIVE FUNCTION.                                    *
* C(IARC)    I*4  - THE COST OF USING THE ARC NUMBER IARC.                *
* CALPHA      R     1-ALPHA = THE COMPLEMENT OF ALPHA                      *
* DEMAND     I*2  - DEMAND AT A QUOTA NODE.  THE DEMAND ALSO DETERMINES *
*                   THE NUMBER OF ARCS GOING FROM EACH QUOTA NODE TO      *
*                   RESPECTIVE QUOTA GROUP NODE.                           *
* CP(IARC)   I*2  - THE DEMAND ALONG ARC NUMBER IARC.                     *
* E2NUM      I*2  - E2 NUMBER. THE E2 NUMBER IS USED IN CONJUNCTION       *
*                   WITH THE FMCC TO DEFINE THE DIFFERENT QUOTAS.         *
* FIT(IARC)  I*2  - FIT LEVEL OF THE IARC-TH MATCH.  FIT IS USED IN       *
*                   CONJUNCTION WITH PCS COST (PCSC) TO FORM A WEIGHTED *
*                   SUM THAT'S USED AS THE COST ON THE SUPPLY-QUOTA ARC *
* FMCC       CHAR - FUTURE MCC.  THE MCC TO WHICH THE INDIVIDUAL IS       *
*                   ON A PARTICULAR ARC.                                   *
* H(INODE)   I*4  - POINTER THAT TELLS WHERE IN THE TAIL LIST TO BEGIN    *
*                   LOOKING FOR THE TAILS ASSOCIATED WITH THE INODE-TH    *
*                   HEAD.                                                  *
* IARC       I*4  - THE INDEX FOR ARCS.                                    *
* IDNUM      I*4  - THE ID NUMBER OF THE PERSON MATCHED ON THE ARC.       *
* INODE      I*4  - THIS IS THE INDEX FOR NODES.                           *
* IQUOTA     I*4  - INDEX USED FOR QUOTAS IN THE GENERATION OF THE        *
*                   SECOND COLUMN OF ARCS (MATCHING QUOTAS TO QUOTA       *
*                   GROUPS.  SERVES AS AN INDEX TO QDEM(I).               *
* ISINKT     I*4  - INDEX FOR SINK NUMBER 1 TAIL                           *
* NNFSTQ     I*4  - NODE NUMBER OF THE FIRST QUOTA IN A QUOTA GROUP       *
* NNLSTQ     I*4  - NODE NUMBER OF THE LAST QUOTA IN A QUOTA GROUP        *
* NUMQ(QGNUM)I*4  - THIS IS THE TOTAL NUMBER OF QUOTAS IN THE QGNUM-TH    *
*                   QUOTA GROUP.                                           *
* NUMQG      I*2  - THIS IS THE TOTAL NUMBER OF QUOTA GROUPS.             *
* OE2NUM     I*2  - OLD E2NUMBER. THE E2NUM OF THE LAST RAW ARC- USED     *
*                   TO TELL IF A NEW QUOTA HAS BEEN BEGUN.                *
* OFMCC      CHAR - OLD FUTURE MCC. THE FMCC OF THE PREVIOUS RAW ARC      *
*                   CARD.  USED TO TELL IF A NEW QUOTA HAS BEEN STARTED *
* OQGRP      I*2  - OLD QUOTA GROUP. THIS IS THE QUOTA GROUP (=SPL) OF    *
*                   THE PREVIOUS RAW ARC CARD. TELLS IF A NEW QUOTA OR    *
*                   QUOTA GROUP HAS BEEN STARTED.                          *
* PCSC(IARC) I*4  - PCS COST OF THE IARC-TH MATCH.  THIS IS USED WITH     *
*                   FIT TO FORM A WEIGHTED COST ON THE SUPPLY-QUOTA ARC *
* QDEM(QNUM) I*2  - DEMAND OF THE QNUM-TH QUOTA                            *
* QDMAND       R  - REAL VARIABLE USED TO ALLOW DIVISION IN THE FORMULA *
*                   USED TO DETERMINE THE SPECIAL COSTS ON THE ARCS       *
*                   FROM THE QUOTAS TO QUOTA GROUPS.                       *
* QGDEM(QGNUM)I*4- DEMAND  IN  THE QGNUM-TH QUOTA GROUP.  THIS IS         *
*                   FOUND BY ADDING ALL THE DEMANDS IN THE QUOTA GROUP. *
* QNUM        I*4 - QUOTA NUMBER. INDEX FOR QUOTAS WHICH GIVES THE        *
*                   CUMULATIVE TOTAL OF QUOTAS IN ALL QUOTA GROUPS        *
* QGNUM       I*4 - QUOTA GROUP NUMBER.  THIS IS THE INDEX FOR QUOTA      *
*                   GROUPS.                                                *
* QGRP(QGNUM)I*2 - THIS GIVES THE SPL VALUE FOR THE QGNUM-TH QUOTA        *
*                   GROUP.                                                 *
* SPL         I*2 - STAFFING PRECEDENCE LEVEL. QUOTA GROUPS ARE DEFINED *
*                   AS SETS OF ARCS WITH THE SAME SPL                      *
* SUPSIZ      I*4 - SUPPLY SIZE. THIS NUMBER REPRESENTS THE TOTAL         *
*                   NUMBER OF INDIVIDUALS (BOTH MOVERS AND NON-MOVERS     *
*                   IN THE PROBLEM. IT IS READ FROM SUP-SIZE DATA A1.     *
* T(IARC)     I*4 - GIVES THE ACTUAL NODE NUMBER FOR THE TAIL OF ARC      *
*                   NUMBER IARC. THIS IS THE TAIL LIST.                    *
* TNUMQ       I*2 - TOTAL NUMBER OF QUOTAS IS ALL QUOTA GROUPS.  THIS     *
*                    EQUALS THE FINAL VALUE OF QNUM.                    *
* TNUMQG      I*2 - TOTAL NUMBER OF QUOTA GROUPS.  THIS IS EQUAL TO THE *
*                    FINAL VALUE OF QGNUM.                              *
* TOTDEM      I*4 - TOTAL DEMAND FOR ALL QUOTA GROUPS (ENTIRE PROBLEM)  *
* UBOUND      I*4 - UPPER BOUND ON THE CAPACITY OF THE ARC BETWEEN      *
*                    SINK1 AND SINK2.  THE VALUE DEFAULTS TO 30000 FOR  *
*                    THE MAX FILL PROBLEM, BUT MAY BE LOWERED BY THE    *
*                    USER TO GIVE GNET THE ABILITY TO IMPROVE THE FIT / *
*                    PCS COST SOLUTION                                  *
* X(INODE)    I*4 - GIVES THE SUPPLY (OR DEMAND IF NEGATIVE)            *
*                    ASSOCIATED WITH NODE NUMBER INODE.                 *
*                                                                       *
*************************************************************************
       IMPLICIT INTEGER (A-Z)
       CHARACTER FMCC,OFMCC
       INTEGER*2 AFLAG,SPL,OQGRP,E2NUM,OE2NUM,DEMAND,QDEM(10000),
      1 QGDEM(200),IDNUM,QGNUM,NUMQG,QNUM,P,D,IT,NSA,A,
      2 NUMQ(200),QGRP(200),TNUMQ,TNUMQG,X(20000),T(90000),FIT(90000)
       INTEGER*4 SUPSIZ,H(20000),C(90000),CP(90000),CPX,U
      1 INODE,IARC,PCSC(90000),NNFSTQ,NNLSTQ,ISA
      2,IQUOTA,ISINKT,TOTDEM,UBOUND,IBIG
       REAL ALPHA,CALPHA,QDMAND
*     THE FIRST STEP IN GENERATING THE ARCS AND COSTS FOR THE CAPACI-   *
* TATED TRANSSHIPMENT PROBLEM IS THE GENERATION OF THE SUPPLY ARCS.     *
* THIS IS DONE BY FIXING H(L) AND X(L) VALUES FOR ALL OF THE SUPPLY     *
* NODES (IE. PEOPLE.)  WE GET THE NUMBER OF SUPPLY ARCS WHICH MUST BE   *
* GENERATED FROM THE FILE SUPSIZ DATA A1 WHICH CONTAINS THE NUMBER OF   *
* INDIVIDUALS WHOSE RECORDS WERE READ OUT OF THE INVENTORY FILE.  WE    *
* THEN SIMPLY LOOP THAT MANY TIMES GENERATING VALUES FOR H(L) AND X(L).*
       READ (30,101) SUPSIZ
101    FORMAT(I5)
****   BEGIN TEST PORTION
       WRITE(32,101) 'SUPSIZ =',SUPSIZ
*****  END TEST PORTION
       DO 10 I = 1,SUPSIZ
          H(I) = 1
          X(I) = 1
10     CONTINUE
*                                                                       *
*     THE NEXT STEP IN THE PROCESS IS GENERATING THE ARCS THAT RUN FROM *
* THE SUPPLY (EACH INDIVIDUAL) TO THE SPECIFIC JOB TYPE, ALSO KNOWN AS  *
* "QUOTA".  EACH QUOTA IS DEFINED BY ALL DEMANDS SHARING THE SAME       *
* E2NUM AND FMCC.  (THAT IS, THE SAME JOB TYPE OR DESCRIPTION AND THE   *
* SAME PLACE.)  THE NODE INDEX IS INODE.  THE ARC INDEX IS IARC.        *
* NOTE THAT THE ARC INDEX BEGINS FROM HERE EQUAL TO 1, BUT THE NODE     *
* INDEX CONTINUES FROM SUPSIZ WHICH IS THE NUMBER OF SUPPLY NODES.      *
*                                                                       *
*     THE ARCS GENERATED BETWEEN SUPPLY AND QUOTAS CONSTITUTE THE FIRST *
* COLUMN OF ARCS IN THE PROBLEM.  FOR EACH OF THESE ARCS WE WILL NEED   *
* TO ASSIGN THE FOLLOWING INDEXED VARIABLES FOR GNET:  H(INODE),        *
* TAIL(IARC), X(INODE), C(IARC), AND CP(IARC).                          *
*                                                                       *
*   INITIALIZE NODE AND ARC INDICES                                     *
       INODE = SUPSIZ + 1
       IARC = 1
*    READ AND SET ALPHA VALUE FOR USE IN FINDING FIT/PCS OBJ F'N WEIGHTS.
       READ (01,102) AFLAG,ALPHA
102    FORMAT(I1,1X,F4.2)
       IF (AFLAG .EQ. 0.0) ALPHA = .99
       CALPHA = 1.0 - ALPHA
*    READ SORTED INFORMATION CONTAINING RAW ARC INFORMATION.
11     READ (31,103,END = 9991) SPL,E2NUM,FMCC,DEMAND,FIT(IARC),
       CPCSC(IARC),IDNUM
103    FORMAT (I2,1X,I4,1X,A3,1X,I3,1X,I2,1X,I5,1X,I5)
*    IF FIRST ARC IN SUPPLY-QUOTA COLUMN THEN ASSIGN CORRECT VALUES
*   AND INITIALIZE COUNTERS AND INDICES.
       IF (IARC .EQ. 1) THEN
*   INITIALIZE THE QUOTA AND QUOTA GROUP INDICES.
           NUMQ(1) = 1
           NUMQG = 1
           QNUM = 1
           QGNUM = 1
           H(INODE) = IARC
           T(IARC) = IDNUM
           X(INODE) = 0
           C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C        CALPHA*PCSC(IARC) + 0.5) + 100
           CP(IARC) = 1
           QGRP(QGNUM) = SPL
           QDEM(QNUM) = DEMAND
           QGDEM(QGNUM) = DEMAND
           OE2NUM = E2NUM
           OFMCC = FMCC
           OQGRP = SPL
*         WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
107    FORMAT(6X,2I6,2X,4I10)
       ELSE IF (IARC .NE. 1) THEN
           IF (FMCC .EQ. OFMCC) THEN
               IF (E2NUM .EQ. OE2NUM) THEN
                   IF (SPL .EQ. OQGRP) THEN
                       T(IARC) = IDNUM
                       C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C                  CALPHA*PCSC(IARC) + 0.5) + 100
                       CP(IARC) = 1
*         WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                   ELSE IF (SPL .NE. OQGRP) THEN
                       INODE = INODE + 1
                       H(INODE) = IARC
                       X(INODE) = 0
                       T(IARC) = IDNUM
                       C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C                  CALPHA*PCSC(IARC) + 0.5) + 100
                       CP(IARC) = 1
                       OQGRP = SPL
                       QNUM = QNUM + 1
                       QDEM(QNUM) = DEMAND
                       QGNUM = QGNUM + 1
                       QGDEM(QGNUM) = DEMAND
                       QGRP(QGNUM) = SPL
                       NUMQ(QGNUM) = 1
*         WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                   ENDIF
               ELSE IF (E2NUM .NE. OE2NUM) THEN
                   IF (SPL .EQ. OQGRP) THEN
                       INODE = INODE + 1
                       X(INODE) = 0
                       QNUM = QNUM + 1
                       NUMQ(QGNUM) = NUMQ(QGNUM) + 1
                       QDEM(QNUM) = DEMAND
                       H(INODE) = IARC
                       T(IARC) = IDNUM
                       C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C                  CALPHA*PCSC(IARC) + 0.5) + 100
                       CP(IARC) = 1
                       OE2NUM = E2NUM
                       QGDEM(QGNUM) = QGDEM(QGNUM) + DEMAND
*         WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                   ELSE IF (SPL .NE. OQGRP) THEN
                       INODE = INODE + 1
                       X(INODE) = 0
                       QNUM = QNUM + 1
                       QDEM(QNUM) = DEMAND
                       H(INODE) = IARC
                       T(IARC) = IDNUM
                       C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C               CALPHA*PCSC(IARC) + 0.5) + 100
                     CP(IARC) = 1
                     OE2NUM = E2NUM
                     OQGRP = SPL
                     QGNUM = QGNUM + 1
                     QGRP(QGNUM) = SPL
                     NUMQ(QGNUM) = 1
                     QGDEM(QGNUM) = DEMAND
*        WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                 ENDIF
              ENDIF
           ELSE IF (FMCC .NE. OFMCC) THEN
              IF (E2NUM .EQ. OE2NUM) THEN
                 IF (SPL .EQ. OQGRP) THEN
                     INODE = INODE + 1
                     X(INODE) = 0
                     QNUM = QNUM + 1
                     NUMQ(QGNUM) = NUMQ(QGNUM) + 1
                     QDEM(QNUM) = DEMAND
                     H(INODE) = IARC
                     T(IARC) = IDNUM
                     C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C               CALPHA*PCSC(IARC) + 0.5) + 100
                     CP(IARC) = 1
                     OFMCC = FMCC
                     QGDEM(QGNUM) = QGDEM(QGNUM) + DEMAND
*        WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                 ELSE IF (SPL .NE. OQGRP) THEN
                     INODE = INODE + 1
                     X(INODE) = 0
                     QNUM = QNUM + 1
                     QDEM(QNUM) = DEMAND
                     H(INODE) = IARC
                     T(IARC) = IDNUM
                     C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C               CALPHA*PCSC(IARC) + 0.5) + 100
                     CP(IARC) = 1
                     OFMCC = FMCC
                     OQGRP = SPL
                     QGNUM = QGNUM + 1
                     QGRP(QGNUM) = SPL
                     NUMQ(QGNUM) = 1
                     QGDEM(QGNUM) = DEMAND
*        WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                 ENDIF
              ELSE IF (E2NUM .NE. OE2NUM) THEN
                 IF (SPL .EQ. OQGRP) THEN
                     INODE = INODE + 1
                     X(INODE) = 0
                     QNUM = QNUM + 1
                     NUMQ(QGNUM) = NUMQ(QGNUM) + 1
                     QDEM(QNUM) = DEMAND
                     H(INODE) = IARC
                     T(IARC) = IDNUM
                     C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C               CALPHA*PCSC(IARC) + 0.5) + 100
                     CP(IARC) = 1
                     OE2NUM = E2NUM
                     OFMCC = FMCC
                     QGDEM(QGNUM) = QGDEM(QGNUM) + DEMAND
*        WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                 ELSE IF (SPL .NE. OQGRP) THEN
                     INODE = INODE + 1
                     X(INODE) = 0
                     QNUM = QNUM + 1
                     QDEM(QNUM) = DEMAND
                     H(INODE) = IARC
                     T(IARC) = IDNUM
                     C(IARC) = INT( ALPHA*(1000*((FIT(IARC)-1)*2)) +
     C               CALPHA*PCSC(IARC) + 0.5) + 100
```

```
                    CP(IARC) = 1
                    OE2NUM = E2NUM
                    OFMCC = FMCC
                    OQGRP = SPL
                    QGNUM = QGNUM + 1
                    QGRP(QGNUM) = SPL
                    NUMQ(QGNUM) = 1
                    QGDEM(QGNUM) = DEMAND
*         WRITE(34,107) IDNUM,INODE,C(IARC),CP(IARC),X(INODE)
                  ENDIF
                ENDIF
              ENDIF
            ENDIF
            IARC = IARC + 1
            GO TO 11
9991      CONTINUE
            INODE = INODE + 1
            TNUMQ = QNUM
            TNUMQG = QGNUM
*      AT THIS POINT, WE HAVE GENERATED THE ARCS IN THE FIRST COLUMN OF  *
* THE CAPACITATED TRANSSHIPMENT PROBLEM FORMULATION.  THE ARCS WE HAVE   *
* MADE GO FROM THE INDIVIDUAL MARINES TO THE QUOTAS TO WHICH EACH OF THE*
* MARINES IS ELIGIBLE TO BE TRANSFERRED.  NEXT, WE MUST MAKE ARCS WHICH *
* CONNECT THE QUOTAS TO THEIR RESPECTIVE QUOTA GROUPS.  SINCE WE WISH TO*
* INCORPORATE A "PROPORTIONALITY OF FILL" CONSTRAINT, WE WILL BE MAKING *
* ADDITIONAL ARCS.  BETWEEN EACH QUOTA AND QUOTA GROUP THERE WILL BE A  *
* NUMBER OF ARCS EQUAL TO THE DEMAND AT EACH QUOTA.  EACH OF THESE ARCS *
* WILL HAVE A COST ON IT WHICH ENFORCES THE PROPORTIONALITY OF FILL     *
* CONSTRAINT (WHICH COULD ALSO BE REGARDED AS MAKING ALL QUOTAS WITHIN A*
* QUOTA GROUP SHARE SHORTAGES WHENEVER POSSIBLE.)  SEE KLINGMAN AND     *
* PHILLIPS FOR A MORE DETAILED EXPLANATION OF THE REASONING BEHIND THE  *
* ADDITIONAL ARCS AND THEIR SPECIALIZED COSTS.                          *
*      WE NOW PROCEED TO GENERATE THE SECOND COLUMN OF ARCS.            *
*                                                                       *
***      INITIALIZE THE INDEX FOR QUOTAS AND THE STARTING QUOTA NUMBER     *
           IQUOTA = 1                                                        *
           NNFSTQ = SUPSIZ + 1                                               *
***      DO FOR EACH QUOTA GROUP
           DO 21 I = 1,TNUMQG
               H(INODE) = IARC
               X(INODE) = 0
               NNLSTQ = NNFSTQ + NUMQ(I) - 1
***      LOOK AT ALL QUOTAS IN THE QUOTA GROUP
               DO 22 J = NNFSTQ,NNLSTQ
***      FOR EACH QUOTA, MAKE QDEM(IQUOTA) ARCS WITH THE APPROPRIATE COST.
                   DO 23 K = 1,QDEM(IQUOTA)
                       QDMAND = QDEM(IQUOTA)
                       T(IARC) = J
                       IF (QDMAND .EQ. 1) THEN
                           C(IARC) = 0
                       ELSE IF (QDMAND .NE. 1) THEN
                           C(IARC) = INT(100*(1/(QDMAND)*(K-0.5)) + 0.5)
                       ENDIF
                       CP(IARC) = 1
                       IARC = IARC + 1
23                 CONTINUE
***      ONCE ALL ARCS ARE MADE FOR THAT QUOTA, INCREMENT IQUOTA TO MARK
*        THE NEXT QUOTA.
                   IQUOTA = IQUOTA + 1
22             CONTINUE
***      ONCE ALL QUOTAS IN A QUOTA GROUP HAVE BEEN TAKEN CARE OF, SET THE
*        STARTING NODE NUMBER OF THE FIRST QUOTA IN THE NEXT QUOTA GROUP
               NNFSTQ = NNLSTQ +1
               INODE = INODE + 1
21       CONTINUE
*      AT THIS POINT, WE HAVE GENERATED THE ARCS CONNECTING THE QUOTAS   **
* TO THE QUOTA GROUPS.  IT REMAINS TO JOIN THE QUOTA GROUPS SINK1 AND   **
* THEN TO JOIN SINK1 TO SINK2.  IN THIS NEXT SECTION OF THE PROGRAM    *
* WE MAKE THE ARCS CONNECTING THE QUOTA GROUPS TO THE FIRST SINK.      *
* WE ATTACH COSTS ON THOSE ARCS WHICH WILL ENFORCE THE CONSTRAINT      *
* THAT, ONCE THE TOTAL NUMBER OF JOBS IS MAXIMIZED, JOBS SHOULD BE     *
* FILLED IN ORDER OF PRECEDENCE OR PRIORITY LEVEL (SPL).               *
      ISINKT = SUPSIZ + TNUMQ + 1
      H(INODE) = IARC
      X(INODE) = 0
         DO 31 I = 1,TNUMQG
            T(IARC) = ISINKT
            CP(IARC) = QGDEM(I)
            IF (QGRP(I) .EQ. 1) THEN
               C(IARC) = -40000
            ELSE IF (QGRP(I) .EQ. 2) THEN
               C(IARC) = -30000
            ELSE IF (QGRP(I) .EQ. 3) THEN
               C(IARC) = -20000
            ELSE IF (QGRP(I) .EQ. 4) THEN
               C(IARC) = -10000
            ELSE IF (QGRP(I) .EQ. 5) THEN
               C(IARC) = 0
            ENDIF
            ISINKT = ISINKT + 1
            IARC = IARC + 1
31       CONTINUE
*    NEXT, GENERATE THE ARC FROM SINK1 TO SINK2.
      INODE = INODE + 1
      H(INODE) =IARC
      TOTDEM = 0
*   FIND THE TOTAL DEMAND FOR ALL QUOTA GROUPS
      DO 32 I = 1,TNUMQG
         TOTDEM = TOTDEM + QGDEM(I)
32    CONTINUE
      X(INODE) = -(TOTDEM)
      T(IARC) = (INODE - 1)
      C(IARC) = 0
*   NOW READ IN THE UPPER BOUND ON THE TOTAL FILL.  NORMALLY WE
* WOULD NOT WANT TO PLACE ANY UPPER BOUND ON FILL SINCE OUR GOAL IS TO
* MAXIMIZE FILL.  THUS, THE UPPER BOUND ON FILL, "UBOUND" DEFAULTS TO AN
* ARBITRARILY HIGH NUMBER (30,000).  HOWEVER, IT IS POSSIBLE THAT THE
* DECISION MAKER WOULD BE WILLING TO TRADE OFF SOME FILL IN ORDER TO
* GAIN AN IMPROVEMENT IN THE FIT OR PCS COST RESULT.  IN ORDER TO PERMIT
* THIS, THE MODEL MAKES PROVISION FOR REDUCING THE FILL TO SOME USER-
* CONTROLLED UPPER BOUND IN ORDER TO GIVE THE SOLVER MORE FLEXIBILITY
* IN IMPROVING THE SOLUTION IN THE OTHER OBJECTIVES.
*   READ IN THE VALUE OF THE UPPER BOUND
      READ(02,106) UBOUND
106   FORMAT(I5)
      CP(IARC) = UBOUND
*        THE FINAL TASK IN COMPLETING THE TRANSSHIPMENT FORMULATION
* FORMATTING FOR GNET IS TO MAKE THE ARTIFICIAL ARCS WHICH GO FROM THE
* SUPPLY TO SINK2.  EACH OF THESE ARCS WILL HAVE A CAPACITY OF ONE AND
* AN ARBITRARILY HIGH COST (= 99999) IN ORDER TO DISCOURAGE GNET FROM
* SENDING ANYONE ALONG THESE ARCS.
*        THE HEAD NODE, H(INODE), AND X(INODE) DO NOT CHANGE, HOWEVER,
* WE MUST ADD TAILS, COSTS AND CAPACITIES BACK TO ALL OF THE SUPPLY
* NODES.
      IARC = IARC + 1
      DO 41 I = 1,SUPSIZ
         T(IARC) = I
         C(IARC) = 99999
         CP(IARC) = 1
         IARC = IARC + 1
41    CONTINUE
*   TO LET GNET KNOW THAT WE HAVE FINISHED, WE MUST GENERATE A FINAL
* HEAD NODE POINTER, H(INODE), WHICH POINTS TO THE TAIL ARRAY AT THE
* FIRST PLACE AFTER THE LAST TAIL.  WE WILL THEN ASSIGN THE VALUE OF
* THE TOTAL NUMBER OF NODES TO THE VARIABLE "M", AND M+1 TO THE VARIABLE
* VARIABLE MP1.
      M = INODE
      INODE = INODE + 1
      MP1 = INODE
      H(INODE) = IARC
************************************************************************
**     AT THIS POINT, THE ARC LIST IS READY TO BE SENT TO A FILE FOR  **
** GNETBX TO READ THE ARCS IN AND SOLVE THE PROBLEM. GNETBX IS CHOSEN **
** OVER GNETX SINCE IT ALREADY CONTAINS A SMALL REPORT WRITER.         **
************************************************************************
* WRITE IN THE NUMBER OF NODES
      WRITE(34,108) M,2
108   FORMAT(2I5)
*     WRITE(34,109)
*109  FORMAT('          FROM   TO         COST       CAPCTY      LOWR-BND
*    CUPR-BND')
* WRITE ALL REAL ARCS EXCEPT THE ONE FROM THE POOL TO THE SINK.
      DO 70 I=(SUPSIZ + 1),(INODE - 2)
          DO 70 J = H(I), H(I+1)-1
              WRITE(34,107) T(J),I,C(J),CP(J),0,0
70    CONTINUE
* NOW WRITE THE LAST REAL ARC; THE ONE WITH THE VARIABLE UPPER BOUND
      WRITE(34,107) (M-1),M,0,UBOUND,0,0
* NOW WRITE THE ARC FROM THE SINK TO THE SUPER-SINK
      WRITE(34,107) M,(M+2),0,SUPSIZ,0,0
* NOW WRITE THE ARCS FROM THE SUPER-SOURCE TO THE SUPPLY NODES
      DO 71 I = 1,SUPSIZ
          WRITE(34,107) (M+1),I,0,1,0,0
71    CONTINUE
* NOW WRITE THE ARC FROM THE SUPER-SOURCE TO THE SUPER-SINK
      WRITE(34,107) (M+1),(M+2),99999,SUPSIZ,0,0

************************************************************************
*     THE NEXT TEST PRINTOUT IS DESIGNED TO CHECK THE VARIABLES THAT  *
* HAVE BEEN GENERATED FOR FITNESS INTO GNETX FORTRAN.  SINCE GNETX IS  *
* NOT USED IN THE PROTOTYPE, THE TEST HAS BEEN COMMENTED OUT.          *
* IT IS NOT DELETED SINCE A FUTURE ENHANCEMENT MAY WISH TO MAKE USE OF *
* THE INCREASED FLEXIBILITY OFFERED BY GNETX.                          *
************************************************************************
***   BEGIN TEST OUTPUT
*     WRITE (32,*) '     M = ',M,'       MP1 = ',MP1
*     WRITE (32,*) 'INODE = ',INODE,'    IARC = ',IARC
*       WRITE(32,104) TNUMQ,TNUMQG
*104  FORMAT ('TNUMQ = ',I2,'.  TNUMQG = ',I2)
*       WRITE(32,*) 'INDEX H(M) X(M) T(N) FIT(N) PCSC(N) C(N) CP(N) QDE
*    CM(I) QGDEM(I) QGRP(I) NUMQ(I)'
*     DO 433 I = 1,60
*        WRITE(32,105)I,H(I),X(I),T(I),FIT(I),PCSC(I),C(I),CP(I)
*    C,QDEM(I),QGDEM(I),QGRP(I),NUMQ(I)
*105  FORMAT(5I5,I10,I6,I5,I6,I8,I7,I8)
*433  CONTINUE
*   END TEST PORTION
*                                                                      *
*   SINCE THE TOTAL NUMBER OF ARCS SOUGHT, REDUCE IARC BY 1            *
      IND = M + 1
      IAD = IARC - 1
      IOD = M
      NSA = M
      ISA = M
      A = M + 1
      IHS = 0
      MAXC = 0
      ISUP = -1
      IBIG = 1000000000
      MBIG = -1
      NNE = -1
      NNS = -1
```

```
        IPG = -1
        NAP = -1
        IPTG = 3
        IOUT = 25
***   BEGIN TEST OUTPUT
*     WRITE(32,791) IND,IAD,IOD,M,H(M),T(M),C(M),CP(M),
*    CX(M)
*791    FORMAT('IND =',I3,' IAD =',I3,' IOD =',I3,' M =',I3,' H(M) =',I3
*    C,' T(M) =',I3,' C(M) =',I3,' CP(M) =',I5,' CX(M) =',I3)
*       WRITE (06,*) 'FINISHED GENERATING ARCS FOR THE NETWORK'
        GOTO 7777
*6666   CALL GNETX (IND,IAD,IOD,M,H,T,C,CP,X,CPX,P,D,IT,U,NSA,ISA,A,IHS,
*     CIBIG,MAXC,ISUP,MBIG,NNE,NNS,IPG,NAP,IPTG,IOUT,IER,ISCALE,IPVT)
7777  CONTINUE
*****   CERTAIN INFORMATION MUST BE SENT TO A FILE FOR USE IN     *****
***** ANALYZING THE FINAL GNET SOLUTION. (FN = NET-INFO DATA)     *****
        WRITE(27,771) TNUMQG,TOTDEM
771   FORMAT(2I10)
        DO 888 JJ = 1,TNUMQG
*     WRITE(06,*)'TNUMQG/JJ/QGRP(JJ)/QGDEM(JJ)',TNUMQG,JJ,QGRP(JJ),QGDEM
*    C(JJ)
           WRITE(27,772) QGRP(JJ),QGDEM(JJ)
772      FORMAT(2I10)
888   CONTINUE
*****    NEXT, SEND THE NUMBER OF QUOTA GROUPS TO A FILE WHICH     *****
***** WILL EXTRACT THE FLOWS FROM EACH OF THE QUOTA GROUPS TO THE *****
***** POOL. (FN = TNUMQG DATA)                                     *****
        WRITE (28,773) TNUMQG
773   FORMAT(I10)
        STOP
        END
```

APPENDIX S
GNETBX FORTRAN

1. PROGRAM TO SOLVE THE CAPACITATED TRANSSHIPMENT PROBLEM

```
*************************************************************************
*                                                                       *
*       * * * *  PROGRAM NAME: GNETBX FORTRAN      * * * *              *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                       *
*                 * * *  OVERVIEW AND PURPOSE   * * *                   *
*                                                                       *
*          GNETBX CONTAINS THE PROGRAM GNETB FORTRAN, AN OPTIMIZATION   *
*  SOFTWARE COPYRIGHTED IN 1975 AND 1983 BY :                           *
*                                                                       *
*    GORDON H. BRADLEY, NAVAL POSTGRADUATE SCHOOL, MONTEREY, CA 93940   *
*    GERALD G. BROWN, NAVAL POSTGRADUATE SCHOOL, MONTEREY, CA 93940     *
*    GLENN W. GRAVES, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90024   *
*                                                                       *
*   FOR INFORMATION CONTACT   GLENN W. GRAVES                           *
*                             3642 SEAHORN DRIVE                        *
*                             MALIBU, CA, 90265 USA                     *
*                                                                       *
*   FOR AN EXPLANATION OF THE DESIGN OF THE PROGRAM, SEE:               *
*                                                                       *
*      BRADLEY, G., BROWN, G., AND GRAVES, G.,                          *
*     "DESIGN AND IMPLEMENTATION OF LARGE SCALE PRIMAL TRANSSHIPMENT    *
*      ALGORITHMS,"                                                     *
*      MANAGEMENT SCIENCE, VOL. 24, NO. 1 (SEPT. 1977), PP.1-34.        *
*                                                                       *
*                                                                       *
*    GNETBX EXTRACTS CERTAIN FLOW VARIABLES FROM THE SOLUTION OF        *
* GNETB AND OUTPUTS THEM TO FILES WHICH ARE USED IN THE PRESENTATION    *
* OF THE SOLUTION TO THE DECISION MAKER.                                *
*                                                                       *
*************************************************************************
```

APPENDIX T
SUMMARY FORTRAN

1. PROGRAM FOR SOLUTION PRESENTATION AND PROBLEM MODIFICATION

```
***********************************************************************
*                                                                     *
*      *  *  *  *   PROGRAM NAME: SUMMARY FORTRAN      *  *  *  *     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                  * * *   OVERVIEW AND PURPOSE   * * *               *
*                                                                     *
*     THIS PROGRAM PRESENTS A SHORT SUMMARY OF THE SOLUTION TO THE     *
* USER AND PERMITS HIM TO MAKE CHANGES IN THE PROBLEM IN THREE WAYS.  *
* FIRST, HE MAY CHANGE THE PRIORITY OF THE FIT AND PCS COST           *
* OBJECTIVES.  SECOND, HE MAY REDUCE THE FILL BELOW THAT ACHIEVED IN  *
* THE SOLUTION IN ORDER TO ALLOW MORE TIES FOR THE IMPROVEMENT OF THE *
* FIT/PCS COST OBJECTIVES.  THIRD, HE MAY CHANGE THE TOUR CONTROL     *
* FACTOR (TCF) BY ADJUSTING A TCF ADJUSTMENT FACTOR.   THESE CHANGES  *
* ARE HANDLED BY THE SUBROUTINES CHGWTS (TO CHANGE THE WEIGHTS OF THE *
* FIT/PCS OBJECTIVES), CHGFIL (TO LOWER THE FILL), AND CHPLCY (TO     *
* CHANGE THE POLICY REGARDING TCF'S.)  THESE ROUTINES CONSIST MOSTLY  *
* OF QUESTION FORMATTING. EXPLANATIONS OF SIGNIFICANT SECTIONS ARE    *
* DOCUMENTED WITHIN THE PROGRAMS.                                     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                  ***   FILE DEFINITIONS   ***                       *
*                                                                     *
*    FILEDEF   FILE IDENTIFICATION              PURPOSE               *
*      01        ALPHA DATA          SETS RELATIVE FIT/PCS COST PRIORTY *
*      02        UPPR-BND DATA       PERMITS LOWERING FILL TO IMPROVE *
*                                    FIT/PCS COST OBJECTIVES          *
*      03        BESTNUM DATA        RECORDS MAX FILL OF ANY SOLUTION *
*      04        TCF-ADJ DATA        CHANGE IN TCF (MONTHS)           *
*      06        TERMINAL SCREEN     DISPLAY OUTPUT TO USER           *
*      30        SUPSIZE DATA        GIVES TOTAL SUPPLY IN INVENTORY  *
*      35        NET-INFO DATA       INFORMATION ON NETWORK STRUCTURE *
*      38        SUMMARY INFO1       INFORMATION ON NETWORK SOLUTION  *
*      42        ALPHAX DATA         TEMP STORES USER INPUT ALPHA     *
*      43        TCFXADJ DATA        TEMP STORES USER INPUT TCF ADJ   *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                     *
*                 ***   DEFINITION OF TERMS   ***                     *
*                                                                     *
*   ADJTCF    ADJUSTMENT TO TCF INPUT BY USER                         *
*   ALPHA     USED TO CHANGE RELATIVE WEIGHTS OF FIT/PCS OBJECTIVES   *
*   ARTFLO    NUMBER OF INDIVIDUALS NOT MATCHED TO BILLETS            *
*   AVEFIT    APPROXIMATION OF THE AVERAGE FIT ACHIEVED IN THE SOLN   *
*   COST      THE TOTAL COST OF THE SOLUTION (USING WEIGHTS AS COSTS) *
*   FITPCS    COMBINED FIT/PCS OBJECTIVES "COST"                      *
*   FLOW      THE FLOW IN EACH SPL                                    *
*   ISPL      INDEX FOR SPL                                           *
*   NEWBND    THE NEW USER-INPUT UPPER BOUND ON FILL                  *
*   NUMBIL    NUMBER OF BILLETS OR DEMANDS IN EACH SPL                *
*   PCSC      AVERAGE PCS COST OF THE SOLUTION.                       *
*   PCTFIL    PERCENTAGE OF AUTH BILLETS (NUMBIL) THAT WERE FILLED    *
*   PCTXCS    PERCENTAGE OF INVENTORY THAT WERE NOT MATCHED           *
*   QGDEM     DEMAND AT EACH QUOTA GROUP                              *
*   QGFLOW    ACTUAL FLOW (= NUMBER OF MATCHES) THROUGH A QUOTA GROUP *
*   QGRP      THE SPL ASSOCIATED WITH A QUOTA GROUP                   *
*   REPLY/2/3 USER RESPONSE VARIABLES                                   *
*   SUPSIZ    TOTAL NUMBER OF PEOPLE IN THE INVENTORY                   *
*   TNUMQG    TOTAL NUMBER OF QUOTA GROUPS                              *
*   TOTDEM    TOTAL DEMAND AT ALL QUOTA GROUPS                          *
*   TPCTF     PERCENT OF BILLETS (DEMAND) FILLED                        *
*   TPCTUT    PERCENTAGE OF OFFICERS ALLOCATED                          *
*   TRUFLO    TOTAL FLOW (FILL) IN ALL QUOTA GROUPS                     *
*                                                                       *
*************************************************************************
       INTEGER TRUFLO,ARTFLO,TNUMQG,QGFLOW(5),TOTDEM,QGRP(5),QGDEM(5)
      C,ISPL,NUMBIL(5),FLOW(5),SUPSIZ,NEWBND ,ADJTCF
       REAL COST,PCTFIL(5),TPCTF,TPCTUT,PCSC,PCTXCS,AVEFIT,ALPHA
       CHARACTER REPLY,REPLY2,REPLY3
* INITIALIZE ALL ARRAY ELEMENTS TO 0
       DO 5 I = 1,5
          QGFLOW(I) = 0
          PCTFIL(I) = 0.0
          QGRP(I) = 0
          QGDEM(I) = 0
          NUMBIL(I) = 0
          FLOW(I) = 0
5     CONTINUE
*     READ IN VALUE FOR LSTBST
       READ (03,187) LSTBST
187   FORMAT(I5)
* READ PRESENT UPPER BOUND FROM UPPR-BND DATA FILE
       READ (02,187) UPRBND
* READ PRESENT ALPHA VALUE FROM ALPHA DATA A1.
       READ (01,188) ALPHA
188   FORMAT(2X,F4.2)
* READ PRESENT TOUR CONTROL ADJUSTMENT FACTOR
       READ (04,189) ADJTCF
189   FORMAT(I2)
* READ IN THE TOTAL NUMBER OF QUOTA GROUPS AND THE TOTAL ASR DEMAND
       READ(35,701) TNUMQG,TOTDEM
701   FORMAT(2I10)
* READ IN THE SPL'S AND THE DEMAND (QGDEM) ASSOCIATED WITH EACH
       DO 10 I = 1,TNUMQG
          READ(35,701) QGRP(I),QGDEM(I)
10    CONTINUE
* READ IN THE TOTAL FEASIBLE FLOW (TRUFLO),ARTIFICIAL FLOW, AND COST
       READ(38,702) TRUFLO,ARTFLO,COST
702   FORMAT(2I10,F12.1)
* READ IN THE ACTUAL FLOW FOR EACH QUOTA GROUP
       DO 20 I = 1,TNUMQG
          READ(38,703) QGFLOW(I)
703     FORMAT(I10)
20    CONTINUE
       READ(30,704) SUPSIZ
704   FORMAT (I5)
* SET VALUES TO BE READ INTO SUMMARY CHART
       ISPL = 1
       DO 40 I = 1,TNUMQG
771     IF (QGRP(I) .EQ. ISPL) THEN
             NUMBIL(ISPL) = QGDEM(I)
             FLOW(ISPL) = QGFLOW(I)
             ISPL = ISPL + 1
          ELSE IF (QGRP(I) .NE. ISPL) THEN
*          NUMBIL(ISPL) = 0
*          FLOW(ISPL) = 0
*          QGFLOW(ISPL) = 0
             ISPL = ISPL + 1
             GOTO 771
          ENDIF
40    CONTINUE
*     WRITE(06,*) I
*     DO 50 J = (I+1),5
*        NUMBIL(J) = 0
*        FLOW(J) = 0
*         QGFLOW(J) = 0
*50     CONTINUE
         DO 60 K = 1,5
60     CONTINUE
* CALCULATE SUMMARY INFORMATION
         DO 70 I = 1,5
            IF (NUMBIL(I) .NE. 0) PCTFIL(I) = REAL(FLOW(I))/REAL(NUMBIL(I))
70     CONTINUE
         TPCTF = REAL(TRUFLO)/REAL(TOTDEM)
         TPCTUT = REAL(TRUFLO)/REAL(SUPSIZ)
         PCTXCS = 0.0
         IF (ARTFLO .NE. 0) PCTXCS = REAL(ARTFLO)/REAL(SUPSIZ)
         PCSC = 1.0
         FITPCS = (((COST-( (ARTFLO*99999)+(FLOW(1)*40000)+(FLOW(2)*30000)
        C+(FLOW(3)*20000)+(FLOW(4)*10000)+(0.5*TRUFLO) ) )/TRUFLO) -100)
         PCSC = FITPCS/(ALPHA-1)
         AVEFIT = FITPCS/ALPHA
* WRITE PRINTOUT OF SUMMARY TO SCREEN
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(C6,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '                         * * *    ****    * * *'
         WRITE(06,*) '    '
         WRITE(06,101)
101    FORMAT (' ',11X,'* * * * *     GNET SUCCESSFULLY TERMINATED     *
        C* * * *')
         WRITE(06,*) '    '
         WRITE(06,*) '                         * * *    ****    * * *'
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '             SCROLL TO NEXT PAGE FOR SOLUTION S
        CUMMARY'
         WRITE(06,*) '    '
         WRITE(06,*) '    '
9998   WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
         WRITE(06,*) '    '
9789   WRITE(06 102)
102    FORMAT   .9X,'SUMMARY OF SOLUTION')
         WRITE(06,*) '    '
         WRITE(06,103)
103    FORMAT(' .______________________ ______________________________
        C__________________________.')
         WRITE(06,104)
104    FORMAT(' |  STAFFING  |  NUMBER  OF    NUMBER OF    PERCENTAGE '
        C AVERAGE   |  AVERAGE  |')
         WRITE(06,105)
105    FORMAT(' | PRECEDENCE |    BILLETS      BILLETS     OF BILLETS |
        C   FIT    |     PCS    |')
         WRITE(06,106)
106    FORMAT(' |    LEVEL      AUTHORIZED      FILLED         FILLED
        C  LEVEL   '    COST
         WRITE(06,107)
107    FORMAT(' |____________ ____________  ____________ ____________ _
        C_____________________
         DO 90 I = 1,5
            WRITE(06,108) I,NUMBIL I  FLOW .  PCTFIL I)
108    FORMAT(' |', I7 , 5X        I8,6X      I8 5X     3X F5.3,4X,'     N/
        CA      |      N/A      .  )
90     CONTINUE
         WRITE(06,107)
*      WRITE(06,116)
116    FORMAT(' |              |              |             |            |
      C       |                 |')
             WRITE(06,109) TOTDEM,TRUFLO,TPCTF,AVEFIT,PCSC
109         FORMAT(' |    TOTALS    |',I8,6X,'|',I8,5X,'|',3X,F5.3,4X,'|',4X,F4.
            C2,3X,'|',3X,F6.1,2X,'|')
             WRITE(06,115)
115         FORMAT(' |_____________________________________________________________
            C_______________________|')
             WRITE(06,*) '    '
             WRITE (06,110) SUPSIZ
110         FORMAT('  TOTAL NUMBER OF OFFICERS IN INVENTORY :',I6)
             WRITE (06,111) TRUFLO
111         FORMAT('  TOTAL NUMBER OF OFFICERS ALLOCATED    :',I6)
             WRITE (06,112) TPCTUT
112         FORMAT('  PERCENTAGE OF OFFICERS ALLOCATED      .',1X,F5.3)
             WRITE (06,113) ARTFLO
113         FORMAT('  NUMBER OF OFFICERS NOT ALLOCATED      :',I6)
             WRITE (06,114) PCTXCS
114         FORMAT('  PERCENTAGE OF OFFICERS NOT ALLOCATED  :',1X,F5.3)
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,*) '    '
             WRITE(06,117)
117         FORMAT('       YOU MAY NOW CHANGE THE STAFFING PROBLEM.    ' )
9997        WRITE(06,118)
118         FORMAT('       PLEASE CHOOSE ONE OF THE FOLLOWING OPTIONS:' )
             WRITE(06,*) '    '
             WRITE(06,119)
119         FORMAT('        ENTER KEY                      IN ORDER TO :')
             WRITE(06,120)
120         FORMAT('        _________  ____________________________________________
            C_____________')
921         WRITE(06,121)
121         FORMAT('             1       CHANGE THE PRIORITY OF THE FIT AND PCS COST
            C OBJECTIVES')
922         WRITE(06,122)
122         FORMAT('             2       ALLOW THE FILL TO BE REDUCED TO IMPROVE FIT
            C OR PCS COST')
923         WRITE(06,123)
123         FORMAT('             3       CHANGE THE TOUR CONTROL FACTORS DETERMINING
            C WHO MAY MOVE')
924         WRITE(06,124)
124         FORMAT('             4       VIEW THE SOLUTION TO THE MOST RECENT PROBLE
            CM')
925         WRITE(06,125)
125         FORMAT('             9       QUIT')
             READ(05,126) REPLY
126         FORMAT (A)
             IF (REPLY .EQ. '9 ') THEN
                WRITE(41,555) UPRBND
555           FORMAT(I5)
                WRITE(42,556) 1,ALPHA
556           FORMAT(I1,1X,F4.2)
                GOTO 9987
             ELSE IF (REPLY .EQ. '4 ') THEN
                  GOTO 9789
             ELSE IF (REPLY .EQ. '3 ') THEN
                  CALL CHPLCY(ADJTCF)
             ELSE IF (REPLY .EQ. '2 ') THEN
                   CALL CHGFIL(TRUFLO,NEWBND,LSTBST,UPRBND)
             ELSE IF (REPLY .EQ. '1 ') THEN
                  CALL CHGWTS(ALPHA,LSTBST)
             ELSE
                WRITE(06,*) 'ENTRY ERROR.  PLEASE RE-TYPE YOUR OPTION CHOICE.'
                GOTO 9997
             ENDIF
* ONCE THE USER HAS MADE ANY DESIRED CHANGES TO THE PROBLEM HE MAY
* ELECT TO RUN IT AGAIN.
9987  WRITE(06,127)
127   FORMAT('  DO YOU WISH TO MAKE ANY MORE CHANGES? (Y/N)' )
        READ(05,126) REPLY2
        IF (REPLY2 .EQ. 'Y') THEN
           GOTO 9997
        ELSE IF (REPLY2 .EQ. 'N') THEN
9988      WRITE (06,128)
128       FORMAT('  DO YOU WANT TO RUN THE MODEL AGAIN? (Y/N)')
              READ(05,126) REPLY3
              IF (REPLY3 .EQ. 'N') THEN
129            FORMAT(I2)
                 GOTO 9999
              ELSE IF (REPLY3 .EQ. 'Y') THEN
                 WRITE(41,555) UPRBND
                 WRITE(42,556) 1,ALPHA
                 GOTO 9999
              ELSE
                 WRITE(06,*) ' INPUT ERROR. PLEASE TRY AGAIN.'
                 GOTO 9988
              ENDIF
        ELSE
           WRITE(06,*) ' INPUT ERROR. PLEASE TRY AGAIN.'
           GOTO 9987
        ENDIF
9999  CONTINUE
        STOP
        END
************************************************************************
*****                     SUBROUTINE CHGFIL                       *****
************************************************************************
*                                                                      *
*          THIS SUBROUTINE PERMITS THE USER TO LOWER THE UPPER BOUND   *
*     ON THE NUMBER OF PEOPLE THAT CAN BE ALLOCATED IN A GIVEN         *
*     PROBLEM SOLUTION.  BY REDUCING THE NUMBER OF BILLETS THAT CAN    *
*     BE FILLED, IT INCREASES THE NUMBER OF TIES AVAILABLE FOR         *
*     IMPROVING THE FIT AND PCS COST IN THE SOLUTION.                  *
*                                                                      *
*                                                                      *
*                     DEFINITION OF VARIABLES                          *
*                                                                      *
*                                                                      *
*                                                                      *
*     INDEX  - INDEX NUMBER ASSIGNED TO FMCC OR PMCC                   *
*     MCC    - MCC CODE WHICH IS MATCHED TO AN INDEX NUMBER IN THE     *
*              MCCNUM DATA FILE                                        *
*                                                                      *
************************************************************************
***************    DECLARE, DIMENSION, AND INITIALIZE    ***************
        SUBROUTINE CHGFIL(TRUFLO,NEWBND,LSTBST,UPRBND)
        INTEGER TRUFLO,UPRBND,LSTBST
        CHARACTER REPLY1,REPLY2,REPLY3
        CHARACTER*6 LBOUND
* READ BEST FILL  SOLUTION SO FAR (= THE UNCONSTRAINED SOLUTION FILL.)
*    IF THE UPPER BOUND IS AT ITS INITIAL VALUE OF 30000, THEN THERE IS
* NO CONSTRAINT ON FILL, AND THE ONLY OPTIONS FOR THE USER ARE TO
* LOWER THE UPPER BOUND OR TO LEAVE IT ALONE.
*    IF, HOWEVER, THE UPRBND HAS BEEN LOWERED (AND NOTE THAT THE ONLY
* PLACE IT CAN BE MOVED TO IS BETWEEN 0 AND THE BEST, UNCONSTRAINED
* FILL VALUE), THEN WE MUST GIVE THE USER THE OPTION OF SETTING A NEW
* VALUE BETWEEN THE LOWER AND UPPER BOUNDS, QUITTING, OR RESETTING THE
* INITIAL UNCONSTRAINED BOUND.

* MESSAGE TO USER IF NO UPPER BOUND HAS BEEN ENFORCED YET
        IF (UPRBND .EQ. 30000) THEN
        WRITE(06,103)
103   FORMAT(
      C ' THE PRESENT SOLUTION MAXIMIZES THE TOTAL FILL IN ALL BILLETS',/
      C ' BEFORE ATTEMPTING TO IMPROVE THE FILL OF A PARTICULAR SPL OR',/
      C ' OBTAIN A BETTER FIT OR PCS COST.  THEREFORE, THE FILL THAT')
       WRITE(06,104) TRUFLO
104   FORMAT(' WAS ACHIEVED IN THE PRESENT SOLUTION, ',I5,', IS THE MAXI
      CMUM')
       WRITE(06,105)
105   FORMAT(
      C ' NUMBER OF BILLETS THAT CAN BE FILLED UNDER THE POLICIES THAT',/
      C ' WERE CONSIDERED.  IT MAY BE POSSIBLE TO IMPROVE THE FIT AND',/
      C ' PCS COST OF THE SOLUTION BY ALLOWING FOR A SLIGHT REDUCTION ',/
      C ' IN THE FILL.  IN ORDER TO DO THIS, WE CAN SET A MINIMUM ',/
      C ' ACCEPTABLE FILL LEVEL WHICH MUST BE ACHIEVED.')
       FLAG = 0
       LBOUND = ' NONE '
       WRITE(06,*) ' '
9002  IF (FLAG .EQ. 1) THEN
       WRITE(06,120) UPRBND,TRUFLO
120   FORMAT('      PRESENT MINIMUM FIT LEVEL:  ',I5,'        CURRENT FIL
      CL: ',I5)
       ELSE IF (FLAG .EQ. 0) THEN
          WRITE(06,121) LBOUND,TRUFLO
121   FORMAT('      PRESENT MINIMUM FIT LEVEL: ',A6,'        CURRENT FILL
      C: ',I5)
       ENDIF
       WRITE(06,106)
106   FORMAT(//,'     ENTER KEY                         IN ORDER TO :')
       WRITE(06,107)
107   FORMAT('
      C____________')
       WRITE(06,108) TRUFLO
108   FORMAT('         1         ENTER A MINIMUM FILL LEVEL THAT IS LESS
      CTHAN ',I5)
       WRITE(06,109)
109   FORMAT('                   WHICH MIGHT IMPROVE THE FIT OR PCS COST.')
       WRITE(06,110) TRUFLO
110   FORMAT('         2         REMOVE OR CHANGE AN OLD BOUND ON FILL.')
       WRITE(06,111) TRUFLO
111   FORMAT('         9         QUIT THIS OPTION AND RETURN TO THE MAIN ME
      CNU WITH  .')
       WRITE(06,123)
123   FORMAT('                   NO ADDITIONAL CHANGES.')
9001  READ(05,112) REPLY1
112   FORMAT(A)
       IF ((REPLY1 .EQ. 'N') .OR. (REPLY1 .EQ. 'D')) THEN
          WRITE(06,*) ' '
          GOTO 9001
       ELSE IF (REPLY1 .EQ. '9') THEN
                WRITE(41,555) UPRBND
                WRITE(42,556) 1,ALPHA
555           FORMAT(I5)
556           FORMAT(I1,1X,F4.2)
          RETURN
       ELSE IF ((REPLY1 .EQ. '2') .OR. (REPLY1 .EQ. '1'))  THEN
9006    WRITE(06,113)
113     FORMAT(//,'     ENTER KEY                         IN ORDER TO :')
          WRITE(06,114)
114    FORMAT('
      C____________')
          WRITE(06,115) TRUFLO
115   FORMAT('         1         PUT A NEW LOWER BOUND ON FILL.')
          WRITE(06,116)
116   FORMAT('         2         REMOVE ALL BOUNDS ON FILL AND RETURN TO TH
      CE LAST MENU.')
          WRITE(06,117)
117   FORMAT('         9         RETURN TO THE LAST MENU WITH NO CHANGES.')
9004    READ(05,112) REPLY3
          IF (REPLY3 .EQ. '9') THEN
             GOTO 9002
          ELSE IF (REPLY3 .EQ. '2') THEN
             WRITE(41,112) 30000
             UPRBND = 30000
```

```
177         FORMAT(I2)
               LBOUND = 'NONE'
               FLAG = 0
               GOTO 9002
            ELSE IF (REPLY3 .EQ. '1') THEN
               WRITE(06,*) '    PLEASE INPUT NEW LOWER BOUND ON FILL.'
9005        WRITE(06,119) LSTBST
119     FORMAT ('    *** REMEMBER, YOUR BOUND SHOULD LIE BETWEEN 0 AND ',
        CI5,' ***')
               READ(05,102) NEWBND
                  IF ((NEWBND .LT. 0) .OR. (NEWBND .GT. LSTBST)) THEN
                     WRITE(06,*) '     * * * ERROR! NUMBER OUT OF BOUNDS.
        CPLEASE TRY AGAIN * * *'
                     GOTO 9005
                  ELSE
                     UPRBND = NEWBND
                     WRITE(41,102) NEWBND
102               FORMAT(I5)
                     WRITE(06,122) UPRBND
125               FORMAT(I2)
122               FORMAT('    NEW BOUND ON FILL = ',I5)
                     FLAG = 1
                     GOTO 9002
                  ENDIF
               ELSE
                  WRITE(06,*) '    ENTRY ERROR. PLEASE TRY AGAIN.'
                     GOTO 9006
               ENDIF
            ELSE
               WRITE(06,*) '    ENTRY ERROR.  PLEASE TRY AGAIN.'
               GOTO 9002
            ENDIF

         ELSE IF (UPRBND .NE. 30000) THEN
            FLAG = 1
* PRINT INTRODUCTION
            WRITE(06,124)
124     FORMAT(
        C ' IT MAY BE POSSIBLE TO IMPROVE THE FIT AND PCS COST OF THE ',/
        C ' SOLUTION BY ALLOWING FOR A SLIGHT REDUCTION IN THE FILL.    ',/
        C ' IN ORDER TO DO THIS, WE CAN SET A MINIMUM ACCEPTABLE LEVEL',/
        C ' OF FILL LEVEL WHICH MUST BE ACHIEVED FIRST.')
            GOTO 9002
         ENDIF
99      RETURN
         END
***************************************************************************
*****                      SUBROUTINE CHGWTS                          *****
***************************************************************************
*                                                                         *
*         THIS SUBROUTINE PERMITS THE USER TO CHANGE THE WEIGHTS ON       *
*     THE OBJECTIVE FUNCTIONS FOR FIT AND PCS COST.                       *
*                                                                         *
*                                                                         *
*                      DEFINITION OF VARIABLES                            *
*                                                                         *
*                                                                         *
*                                                                         *
*     INDEX  - INDEX NUMBER ASSIGNED TO FMCC OR PMCC                      *
*     MCC    - MCC CODE WHICH IS MATCHED TO AN INDEX NUMBER IN THE        *
*              MCCNUM DATA FILE                                           *
*                                                                         *
***************************************************************************
***************    DECLARE, DIMENSION, AND INITIALIZE     ***************
         SUBROUTINE CHGWTS(ALPHA,LSTBST)
         INTEGER TRUFLO,UPRBND,LSTBST
         REAL ALPHA
         CHARACTER REPLY1,REPLY2,REPLY3
```

```
*     IF THE UPPER BOUND IS AT ITS INITIAL VALUE OF 30000, THEN THERE IS
* NO CONSTRAINT ON FILL, AND THE ONLY OPTIONS FOR THE USER ARE TO
* LOWER THE UPPER BOUND OR TO LEAVE IT ALONE.
*     IF, HOWEVER, THE UPRBND HAS BEEN LOWERED (AND NOTE THAT THE ONLY
* PLACE IT CAN BE MOVED TO IS BETWEEN 0 AND THE BEST, UNCONSTRAINED
* FILL VALUE), THEN WE MUST GIVE THE USER THE OPTION OF SETTING A NEW
* VALUE BETWEEN THE LOWER AND UPPER BOUNDS, QUITTING, OR RESETTING THE
* INITIAL UNCONSTRAINED BOUND.
* MESSAGE TO USER IF NO UPPER BOUND HAS BEEN ENFORCED YET
        IF (ALPHA .EQ. .99) THEN
        WRITE(06,103)
103    FORMAT(
       C ' THE PRESENT FORMULATION WILL SOLVE FOR FIT FIRST, BEFORE IT ',/
       C ' ATTEMPTS TO MINIMIZE PCS COST. ',/ )
        ELSE IF (ALPHA .EQ. .01) THEN
        WRITE(06,104)
104    FORMAT(
       C ' THE PRESENT FORMULATION WILL MINIMIZE PCS COST FIRST, BEFORE ',
       C ' ATTEMPTING TO MAXIMIZE FIT. ',/)
        ELSE IF (ALPHA .EQ. .50) THEN
        WRITE(06,105)
105    FORMAT(
       C ' THE PRESENT FORMULATION WEIGHTS THE FIT AND PCS COST ',/
       C ' OBJECTIVES EQUALLY. ',/ )
        ENDIF
        WRITE(06,133)
133    FORMAT(
       C ' PLEASE CHOOSE ONE OF THE FOLLOWING OPTIONS: ',/)
7777   WRITE(06,106)
106    FORMAT(//,'    ENTER KEY                        IN ORDER TO :')
        WRITE(06,107)
107    FORMAT('        _________     _____________________________________
       C_____________')
        WRITE(06,108)
108    FORMAT('         1         SOLVE FIT FIRST BEFORE MINIMIZING PCS COST
       C ')
        WRITE(06,110)
110    FORMAT('         2         MINIMIZE PCS COST BEFORE MAXIMIZING FIT.')
        WRITE(06,111)
111    FORMAT('         3         WEIGHT BOTH FIT AND COST OBJECTIVES EQUALL
       CY')
        WRITE(06,112)
112    FORMAT('         9         QUIT WITHOUT CHANGING PRESENT ORDERING OF
       COBJECTIVES.')
9001   READ(05,113) REPLY1
113    FORMAT(A)
        IF (REPLY1 .EQ. '1') THEN
           WRITE(42,101) 1,'0.99'
101    FORMAT(I1,1X,A4)
           RETURN
        ELSE IF (REPLY1 .EQ. '2') THEN
           WRITE(42,101) 1,'0.01'
           RETURN
        ELSE IF (REPLY1 .EQ. '3') THEN
           WRITE(42,101) 1,'0.50'
           RETURN
        ELSE IF (REPLY1 .EQ. '9') THEN
           RETURN
        ELSE
           WRITE (06,*) '    ENTRY ERROR. PLEASE ANSWER AGAIN.'
           GOTO 7777
        ENDIF
99     RETURN
        END
************************************************************************
*****                    SUBROUTINE CHPLCY                        *****
************************************************************************
*                                                                      *
*        THIS SUBROUTINE PERMITS THE USER TO CHANGE THE TOUR CONTROL  *
*     FACTORS FOR ALL BILLETS BY ADJUSTING A COEFFICIENT THAT IS      *
*     APPLIED TO ALL BILLETS EQUALLY.                                 *
*                                                                     *
*                        DEFINITION OF VARIABLES                      *
*                                                                     *
*                                                                     *
*     ADJTCF - ADJUSTMENT TO TOUR CONTROL FACTOR (IN MONTHS)          *
*     INDEX  - INDEX NUMBER ASSIGNED TO FMCC OR PMCC                  *
*     MCC    - MCC CODE WHICH IS MATCHED TO AN INDEX NUMBER IN THE    *
*              MCCNUM DATA FILE                                       *
*                                                                     *
***********************************************************************
***************    DECLARE, DIMENSION, AND INITIALIZE    ***************
       SUBROUTINE CHPLCY(ADJTCF)
       INTEGER TRUFLO,UPRBND,LSTBST,ADJTCF,REPLY1
       REAL ALPHA
       CHARACTER REPLY2,REPLY3
* MESSAGE TO USER IF NO ADJUSTMENT IS PRESENTLY IN EFFECT.
       IF (ADJTCF .EQ. 0) THEN
       WRITE(06,103)
103   FORMAT(
      C ' THE PRESENT FORMULATION USES THE NORMAL TOUR CONTROL FACTORS',/
      C ' SET BY MARINE CORPS ORDER. ',/ )
       ELSE IF (ADJTCF .LT. 0) THEN
       WRITE(06,104) ADJTCF
104   FORMAT(
      C ' THE PRESENT FORMULATION REDUCES THE TOUR CONTROL FACTORS AT ',/
      C ' ALL BILLETS BY ',I2,' MONTHS.'/)
       ELSE
       WRITE(06,105) ADJTCF
105   FORMAT(
      C ' THE PRESENT FORMULATION RAISES THE TOUR CONTROL FACTORS AT',/
      C ' ALL BILLETS BY ',I2,' MONTHS.'/)
       ENDIF
5432  WRITE(06,133)
133   FORMAT(
      C ' PLEASE ENTER THE DESIRED ADJUSTMENT FACTOR (IN MONTHS) ',/)
       WRITE(06,134)
134   FORMAT(
      C ' *** NOTE: YOU MUST CHOOSE A NUMBER BETWEEN -60 AND +60 ***',/)
9001  READ(05,113) REPLY1
113   FORMAT(I2)
       IF (REPLY1 .LT. -60) THEN
          WRITE(06,101)
101   FORMAT('  * * * *     ERROR  -  NUMBER OUT OF BOUNDS    * * * *')
          WRITE(06,134)
          GOTO 5432
       ELSE IF (REPLY1 .GT. 60) THEN
          WRITE(06,101)
          WRITE(06,134)
          GOTO 5432
       ELSE
          WRITE(43,135) REPLY1
135   FORMAT(I2)
          RETURN
       ENDIF
99    RETURN
       END
```

APPENDIX U
CHG-DATA SAS

1. PROGRAM TO UPDATE USER CONTROLLED FILES

```
************************************************************************
*                                                                      *
*      *  *  *  *   PROGRAM NAME: CHG-DATA SAS        *  *  *  *       *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*                                                                      *
*                * * *   OVERVIEW AND PURPOSE    * * *                 *
*                                                                      *
*    THIS PROGRAM TAKES THE USER-INPUT CHANGES AND UPDATES THE FILES   *
* WHICH CONTROL THE NEXT FORMULATION OF THE PROBLEM.  ADDITIONALLY, IT *
* RE-COPIES USMC NONMSUP WHICH MAY BE DAMAGED IN THE MATCHING PROGRAM. *
*    AFTER THIS PROGRAM IS RUN, THE THESIS EXEC PROGRAM ROUTES         *
* PERFORMS THE NEXT ACTION INDICATED BY THE USER.  THIS MAY INCLUDE    *
* RESOLVING USING THE NEW WEIGHTS OR BOUNDS, OR RE-STARTING THE ENTIRE *
* PROGRAM.                                                             *
*      THE PROGRAM IS SELF DOCUMENTING.                                *
*                                                                      *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
  INPUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
         EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
         EDA2 $65-68 OFFTYP $70-77 COSTCTR $79-80;
*                                                                      *
*               ***   DEFINITION OF TERMS   ***                        *
*                                                                      *
*   ALPHA      NEW VALUE OF ALPHA                                      *
*   COSTCTR    COST CENTER CODE INDEX                                  *
*   EDA2       EXPECTED DATE OF ARRIVAL AT A FUTURE MCC                *
*   EXP        EXPERIENCE CODE                                         *
*   FIX        INDICATES IF INDIVIDUAL IS FIXED OR FREE                *
*   GRD        GRADE OR RANK                                           *
*   IDNUM      INDEX OF THE INDIVIDUAL IN THE INVENTORY LIST           *
*   MCC        MONITORED COMMAND CODE  (LOCATION)                      *
*   MOS        MILITARY OCCUPATION SPECIALTY                           *
*   MOS2       FIRST ADDITIONAL MOS                                    *
*   MOS3       SECOND ADDITIONAL MOS                                   *
*   MOSNUM     INDEX OF THE MOS AMONG ALL OTHER MOS'S                  *
*   OFFTYP     OFFICER TYPE CLASSIFICATION                             *
*   TCFADJ     NEW TCF ADJUSTMENT FACTOR                               *
*    LDGRD     THE PREVIOUS GRADE THAT WAS LOOKED AT IN THE LIST       *
*   UPBOUND    NEW UPPER BOUND ON FILL                                 *
*   SEX        SEX CODE                                                *
*   SEARCH     ARRAY VARIABLE NAME CONTAINING START & ENDING POINTS    *
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * **
*  *  *  *  *  *  *  *  *   FILE DEFINITION  *  *  *  *  *  *  *  *  *  *
CMS FILEDEF FIN1 DISK UPPERBND DATA A;
CMS FILEDEF FIN2 DISK ALPHAX DATA A ;
CMS FILEDEF FIN3 DISK EXTRA FIXDFILE A;
CMS FILEDEF FIN4 DISK TCFXADJ DATA A;
CMS FILEDEF FOUT1 DISK UPPR-BND DATA A;
CMS FILEDEF FOUT2 DISK ALPHA DATA A ;
CMS FILEDEF FOUT3 DISK USMC NONMSUP A;
CMS FILEDEF FOUT4 DISK TCF-ADJ DATA A;
************************************************************************

OPTIONS LINESIZE = 80;

*  UPDATE UPPER BOUND ON FILL;
DATA DONE;
```

```
    INFILE FIN1;
    INPUT UPBOUND 1-5;
DATA _NULL_;
    SET DONE;
    IF UPBOUND = . THEN UPBOUND = 30000;
        FILE FOUT1;
            PUT UPBOUND 1-5;
* UPDATE ALPHA VALUE;
DATA DTWO;
    INFILE FIN2;
    INPUT ALPHA $3-6;
DATA _NULL_;
    SET DTWO;
    FLAG = 1;
    IF ALPHA = '.' THEN ALPHA = '0.99';
    IF ALPHA = ' ' THEN ALPHA = '0.99';
        FILE FOUT2;
        PUT FLAG 1 ALPHA $3-6;
* REPAIR USMC NONMSUP FILE;
DATA _NULL_;
INFILE FIN3;
   INPUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           EDA2 $65-68 OFFTYP $70-77 COSTCTR $79-80;
   FILE FOUT3;
     PUT MOS $1-4 GRD $6-7 MOS2 $9-12 MOS3 $14-17 MCC $19-21
           EXP $23 SEX $25 LDO $27 FIX 29 IDNUM 31-35 MOSNUM 37-39
           EDA2 $65-68 OFFTYP $70-77 COSTCTR $79-80;
* UPDATE TCF-ADJ DATA FILE;
DATA DTHREE;
    INFILE FIN4;
    INPUT TCFADJ $ 1-5;
DATA _NULL_;
    SET DTHREE;
    IF TCFADJ = '.' THEN TCFADJ = '    0';
    IF TCFADJ = ' ' THEN TCFADJ = '    0';
        FILE FOUT4;
            PUT TCFADJ 1-5;
```

LIST OF REFERENCES

1. Bradley, G., Brown, G., and Graves, G., "Design and Implementation of Large Scale Primal Transshipment Algorithms," *Management Science*, v. 24, No.1, pp. 1-34, Sept. 1977.

2. no author, *Officer Staffing Goal Model (OSGM) Users Guide*, Decision Systems Associates, Inc., Rockville, MD, September 1983.

3. Rosenthal, R., E., "Principles of Multiobjective Optimization," *Decision Sciences*, v.16, No.2, pp 133-152, Spring, 1985.

4. Klingman, D., Mead, M., and Phillips, N. V., "Network Optimization Models for Military Manpower Planning," *Operational Research*, pp. 786-800, 1984.

5. Klingman, D., Mead, M., and Phillips, N. V., "Topological and Computational Aspects of Preemptive Multicriteria Military Personnel Assignment Problems", *Management Science*, v. 30, No. 11, pp. 1362-1375, Nov. 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93943	1
4.	Professor Paul R. Milch Code 55Mh Department of Operations Research Naval Postgraduate School Monterey, CA 93943	5
5.	Professor Gordon H. Bradley Code 55BB Department of Computer Science Naval Postgraduate School Monterey, CA 93943	5
6.	LTC G.C. Axtell Code MPI-40 Headquarters United States Marine Corps Washington, D.C. 20380-0001	1
7.	MAJ D. Hundley Code MMOA-3 Headquarters United States Marine Corps Washington, D.C. 20380-0001	2
9.	CAPT Philip J. Exner 400 Rambler Rd. Belair, MD 21014	4